VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Software Factory - vývoj webových aplikací - komponenta kreditní platby

Software Factory - Web Application Development - Credit Payment Component

2014 Ondřej Gavenda

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Ondřej Gavenda**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Software Factory - vývoj webových aplikací - komponenta kreditní platby
Software Factory - Web Application Development - Credit Payment Component

Zásady pro vypracování:

Cílem této diplomové práce je podílet se na činnostech a rozvoji laboratoře - Software Factory na katedře Informatiky FEI. Dlouhodobým cílem této laboratoře je vytvořit a rozvíjet prostředí pro výuku, výzkum a vývoj softwarového inženýrství, díky kterému bude možné simulovat a sledovat reálný vývoj aplikací, testovat různé procesy a postupy apod.

Konkrétním cílem této diplomové práce je zaměřit se na vývoj webových aplikací. Vyvinout softwarovou komponentu pro kreditní systém plateb za položky na webu. Tato komponenta pak bude začlenitelná do webových aplikací, které budou vyžadovat platby za položky na webu.
1. Rešerše současného stavu.
2. Návrh a implementace komponenty pro platby pomocí kreditu.
3. Návrh a popis rozhraní pro zapojení do webových aplikací.
4. Otestování komponenty v reálné aplikaci.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Svatopluk Štolfa, Ph.D.**

Datum zadání: 16.11.2012
Datum odevzdání: 07.05.2014

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky 1 OSTRAVA

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 2. května 2014 .....

Tímto bych chtěl velmi poděkovat svému vedoucímu, Ing. S. Štolfovi, Ph.D., za jeho trpělivost a podmětné rady, kterými mi pomohl při zpracování této diplomové práce.

Abstrakt

V této diplomové práci bylo mým cílem vyvinout softwarovou komponentu pro kreditní systém plateb. Dalšími cíli této diplomové práce bylo kromě založení a rozvoje softwarové laboratoře také vytvoření centrálního systému, jakožto centrálního uzlu všech komponent a také webové aplikace, na které tyto komponenty budou otestovány.

Z obecného hlediska jsou v první fázi diplomové práce popsány technologie, které byly pro vývoj využity. Dále zde analyzuji současný stav, týkající se platebních systémů.

Další část této diplomové práce je věnována popisu založení softwarové laboratoře, na kterou navazuje hlavní obsah práce. Jedná se o kapitolu věnovanou kreditnímu systému. Kromě kreditního systému je zde obecně popsán centrální systém a způsob, jakým byl kreditní systém testován ve webové aplikaci.

V závěru práce zhodnocuji dosažené výsledky a také další postup vývoje kreditního i centrálního systému.

Klíčová slova: diplomová práce, PHP, programování, Symfony2, webová aplikace, framework, Doctrine, MySQL, databáze, jQuery, PayPal

Abstract

The main goal in my Thesis was to create a software component for the credit payments. My additional goal was to create a central system as the central hub for all components.

The first phase of the Thesis includes the description of the technology that was used during the development. After, I have analyzed the current situation of using credit card systems.

Next part is devoted to a description of establishing the software laboratory that is followed by the main topic of the work – credit system. In addition to the credit system, there is a general description of the central system and the way that was used to test the credit system in the web-based application.

In closing, I have analyzed the achieved results together with additional procedures of the credit system development and the central system development.

Keywords: master thesis, PHP, programming, Symfony2, web application, framework, Doctrine, MySQL, database, jQuery, PayPal

Seznam použitých zkratek a symbolů

API	–	Application Programming Interface
CSRF	–	Cross-Site Request Forgery
ČNB	–	Česká Národní Banka
ČSOB	–	Československá Obchodní Banka
HTML	–	HyperText Markup Language
HTTP	–	Hypertext Transfer Protocol
HTTPS	–	Hypertext Transfer Protocol Secure
IDE	–	Integrated Development Environment
IPN	–	Instant Payment Notification
KB	–	Komerční Banka
ORM	–	Object-Relational Mapper
PDT	–	PHP Development Tool
PHP	–	PHP: Hypertext Preprocessor
RPC	–	Remote Procedure Call
SMTP	–	Simple Mail Transfer Protocol
SOAP	–	Simple Object Access Protocol
SSL	–	Secure Sockets Layer
TCP	–	Transmission Control Protocol
UDP	–	User Datagram Protocol
UI	–	User Interface
URL	–	Uniform Resource Locator
VÚ	–	Virtuální Účet
WSDL	–	Web Services Description Language
XML	–	Extensible Markup Language
XSD	–	XML Schema Definition

Obsah

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Asi každá větší IT firma/společnost/korporace zabývající se vývojem softwaru, má pro své projekty vytvořeny týmy, které na nich pracují. V týmu by měla fungovat harmonie, měly by existovat určitá pravidla, podle kterých se tým bude řídit a bude je poctivě plnit, popř. se domluvit na konvencích, které bude tým dodržovat. Především v průběhu implementační fáze vývoje softwaru nám určitá pravidla a konvence zajistí to, že kód bude sjednocený ve všech dílčích částech, na kterých se podílí více členů týmu ať se jedná o programátora, kodéra nebo například grafika. V případě, že by na dané aplikaci pracovali stále stejní lidé a měli by na starost neustále jen svou část, nějaká pravidla a konvence by se možná ani závádět nemusely. Bohužel realita je úplně jiná a členové týmu se mohou měnit. Mohou být přesunuti na jinou pozici v týmu, s čímž se můžeme v praxi běžně setkávat. Tyto změny, popř. přesuny v týmu by mohly způsobit situace, kdy by si například programátor musel zvykat na nový styl psaného kódu a celkově se v něm znova zorientovat, čímž by se jeho práce výrazně zpomalila.

Nejprve bych zmínil sekundární cíl mé diplomové práce, čímž bylo založení softwarové laboratoře, která by měla simulovat a sledovat reálný vývoj aplikací, testovat různé procesy, postupy apod. Jelikož softwarová laboratoř bude simulovat vývoj softwaru v týmech, prvotním cílem bylo vytvořit pracovní prostředí, dohodnout se na konvencích popř. pravidlech, která budeme dodržovat. Na základě této softwarové laboratoře, bylo pak hlavním cílem jednotlivých členů týmů, vyvinout několik webových popř. mobilních aplikací a komponent, které byly v našem případě brány i jako webové služby.

Mým konkrétním a primárním cílem diplomové práce bylo kromě spolupráce na zprovoznění softwarové laboratoře, vytvoření softwarové komponenty pro systém plateb za položky na webu. Komponentu bude možné získat jako balíček, ale především bude přístupná jako webová služba. Získání kreditního systému ve formě webové služby nese několik značných výhod, kterými se od možnosti zakoupení balíčku liší.

Uvedl bych 2 hlavní výhody. Jako první je napojení plateb a vše co s nimi bude souviset na námi vytvořený centrální systém umožňující kompletní správu kreditního systému. Jako druhou výhodu bych pak uvedl přístup do uživatelského rozhraní, které bude sloužit především ke správě poplatků a plateb včetně transakční historie.

Hlavním záměrem vytvoření této kopmponenty bylo usnadnění a urychlení platebního procesu, který bývá ve většině webových aplikací někdy až příliš zdlouhavý a navíc bývá uživatel přesměrován do platebního systému, kde tuto platbu může teprve provést. Mezi další důvody pak patří zakomponování více měn pod jeden uživatelský účet ve formě více virtuálních účtů. Bude se jednat o vytvoření hlavního jádra platebního systému, který bude v dalších fázích rozšiřován o novou funkcionalitu. Systém by měl být v určitých bodech napodobeninou stávajících platebních bran (systémů). Důvodem, proč se takový systém vytváří je mimo jiné především možnost konkurence schopnosti týkající se požadovaných poplatků a poskytování především nejdůležitějších prvků k provádění plateb a jejich správě.

V rámci softwarové laboratoře je tedy cílem vytvořit si vlastní platební systém, který je možno nakonfigurovat podle vlastních představ.

2 Technologie

V této sekci blíže popíši technologie, které byly ke splnění cílů mé diplomové práce využity. V první řadě je potřeba zmínit, že kreditní systém je implementován využitím skriptovacího jazyka PHP ve spojení s frameworkem Symfony 2. Jako databázový systém byl pak zvolen MySQL, který je plně podporován ORM frameworkem Doctrine, jenž je součástí standardní edice fameworku Symfony 2. Ve standardní edici frameworku Symfony 2 je taktéž šablonovací systém TWIG, který byl v kreditním systému využit.

Jak jsem již zmínil v úvodu, kreditní systém bude přístupný taktéž jako webová služba. Ke komunikaci s touto webovou službou pak slouží protokol SOAP s využitím popisovacího jazyka WSDL.

2.1 PHP

PHP [1], což je rekurzivní zkratka pro PHP: Hypertext Preprocessor, byl veřejně vydán v červnu 1995 jehož autorem je Rasmus Lerdorf [2]. Jedná se o skriptovací jazyk určený především pro programování dynamických webových aplikací s možnou integrací přímo do HTML kódu. Veškerý kód napsaný v PHP je zpracováván na straně serveru narozdíl například od javasriptu, který je zpracováván na straně klienta. Velkou výhodou skriptovacího jazyka PHP je především to, že pro nově příchozí vývojáře je tento jazyk jednoduchý a křivka rychlosti učení tohoto jazyka bývá často velmi strmá. Taktéž ale PHP nabízí mnoho pokročilých vlastností jejichž složitost se pohybuje na vyšší úrovni.

Poslední verze tohoto jazyka je 5.5.11. Stále se však běžně používá verze 5.3.x, která je nadále podporovaná. Jazyk PHP je podporován mnoha frameworky, mezi které bych zařadil Nette jako nejpoužívanější český produkt s rozsáhlou komunitou a jako zahraniční pak Zend, CakePHP a Symfony (viz kapitola 2.3), který byl k implementaci kreditního systému využit.

Výhody PHP

- Zaměření především na dynamické webové aplikace.
- Nativní podpora databázových systému MySQL (viz kapitola 2.2, PostgreSQL aj.
- Strmá křivka učení.
- Velké množství nabízených hostingů (včetně bezplatných možností).

Nevýhody PHP

- Absence ladících nátrojů v nativním stavu.
- Nekonzistentní názvy funkcí.

2.2 MySQL

[3] Jeden z nejpoužívanějších databázových systémů pro jazyk PHP, který spadá pod společnost Sun Microsystems. Tento systém je možné nainstalovat mimo jiné na platformách Linux a Microsoft Windows. Jedná se tedy o multiplatformní systém, se kterým je možné komunikovat pomocí jazyka SQL. V kombinaci s linuxem, PHP a apachem tvoří velmi dobrý základ pro webový server, kde tato kombinace bývá často nazývána zkratkou LAMP. Databázový systém MySQL používá několik typů uložišť dat. Mezi nejznámější a nejčastěji používané patří InnoDB a MyISAM, kde první zmíněný nyní bývá v různých nástrojích často zvolen jako výchozí. InnoDB nabízí možnosti pro obnovu databáze v případě pádu. MyISAM, který by měl být o něco rychlejší však tuto možnost nemá, což je asi hlavním důvodem, proč většina uživatelů MySQL přechází právě na InnoDB.

Pro správu MySQL databází se v současnosti používají nátroje PhpMyAdmin nebo Adminer.

2.3 Symfony 2

Veškeré webové systémy, aplikace a komponenty byly v rámci softwarové laboratoře vyvíjeny na základě frameworku Symfony 2.

Symfony [4] je open source PHP framework vydaný pod licencí MIT. Jedná se o produkt agentury SensioLabs, jejímž původním záměrem bylo využití frameworku pro své vlastní účelu. Symfony se postupem času rozrostlo do takových rozměrů, že se agentura SensioLabs, jejíž obrat za poslední 3 roky (2009-2011) vzrostl o 4,5 milionu eur, rozhodla tento framework v roce 2005 zveřejnit v podobě první verze. Konec vývoje první verze byl pak v listopadu 2012. Před ukončením vývoje verze 1 byla v srpnu 2011 zveřejněna verze druhá, která bude hlavním předmětem této kapitoly.

Výše zmíněný obrat SensioLabs zde uvádím především z toho důvodu, že se vývojáři popřípadě společnosti zakládající se na Symfony nemusí s největší pravděpodobností obávat případného ukončení vývoje tohoto frameworku. Alespoň tedy z finančního hlediska má Symfony vytvořeny pevné základy. Dalším významným projektem této agentury je šablonovací systém TWIG, který je ve standardním balíčku Symfony integrován. Blíže pak tento šablonovací systém popíši v kapitole 2.4 na straně 8.

Kolem Symfony 2 se vede spousta sporů, zda-li se jedná o framework postavený na architektuře MVC či nikoliv. Hlavní vývojář tohoto fameworku (Fabien Potencier) ve svém článku [5] popisuje Symfony jako HTTP framework, nebo-li Request/Response framework. Symfony je však velice flexibilní framework, který je možné přízpůsobit podle vlastních potřeb. Ve standardní edici je možné v adresářové struktuře najít pohledy (views) a řadiče (controllery), kde řadiče jsou zodpovědné za zpracování příchozího požadavku a vrácení odpovědi. Model však není pevně stanoven, a především proto se nedá o frameworku tvrdit, že se jedná o MVC architekturu. Symfony tak dává volnou volbu uživatelům, jakým způsobem si svou modelovou část vytvoří.

Pro potřeby softwarové laboratoře konkrétně pro vývoj komponent má Symfony 2 vyhovující vlastnost a to takovou, že je postaveno na základě modulárního vývoje neboli na systému bundlů (pluginů). Vše od jádra až po vyvíjenou aplikaci je právě v bundlech,

které jsou od zbytku frameworku odděleny jmennými prostory. Je tak možné tyto bundly uchopit a jednoduše zprovoznit v jiné Symfony aplikaci.

2.4 TWIG

[6] Jedná se o moderní šablonovací systém, určený pro jazyk PHP. O tomto systému se zmiňuji především z toho důvodu, že je integrován ve standardní edici frameworku Symfony a je tak v něm využit jako defaultní nátroj pro práci se šablonami. Stejně jako Symfony je tento systém produktem agentury SensioLabs.

TWIG nabízí spoustu tagů, filtrů, funkcí díky kterým je možné přímo v šabloně upravovat hodnoty proměnných nebo také vytvářet libovolnou logiku například ve formě podmínek a cyklů. Šablonovací systém TWIG je taktéž velmi flexibilní a mimo defaultní tagy, funkce a filtry, které jsou zveřejněny v rozsáhlé dokumentaci [7], nabízí možnost vytvoření svých vlastních.

Samotný jazyk PHP se dá taktéž použít jako šablonovací systém, nicméně kód šablon je v tomto případě velmi nepřehledný, čemuž se snaží TWIG předejít. Kromě nesrovnatelného zpřehlednění, se taktéž TWIG zaměřuje na co největší úsporu místa co se týče psaného kódu.

2.5 Doctrine 2

Doctrine 2 [8] je nástroj pro mapování objektů na relační databázi (ORM) pro jazyk PHP 5.3 a vyšší. Doctrine 2 se od první verze velice liší. Doctrine 2 je vytvořeno jako čisté ORM, zatímco první verze obsahovala i jiné vlastnosti, které do ORM vrstvy nepatří, což způsobovalo méně přehledný a složitý kód. Cílem Doctrine 2 tedy bylo odebrat všechny nadbytečné vlastnosti a vytvořit tak čistý ORM framework. [9]

Tento framework slouží především pro usnadnění práce s databází. K určitým operacím využívá příkazovou konzoli. Mezi operace proveditelné pomocí konzole patří mimo jiné například generování entit z již existující databáze nebo naopak vytvoření databázových tabulek z předvytvořeným entitních tříd.

Pro všechny operace manipulující s databází (INSERT, UPDATE, DELETE, SELECT), hraje významnou roli tzv. EntityManager, pomocí kterého jsou všechny tyto operace vykonávány.

Doctrine 2 se skládá ze tří vrstev:

Common Jedná se o první vrstvu, která není nijak závislá na ostatních vrstvách. Tato vrstva obsahuje základní třídy a knihovny, manipulující mimo jiné například s anotacemi, cachováním nebo událostmi.

DBAL Abstraktní databázová vrstva, závislá na první vrstvě, ale nikoliv na třetí (poslední). Cílem této vrstvy je pak sjednocení komunikace aplikace s různými typy databází

ORM Jedná se o nejvyšší vrstvu, která je plně závislá na výšše zmíněných vrstvách a plní hlavní cíl Doctrine, což je právě mapování objektů na relační databázi.

2.6 SOAP

Simple object access protocol (SOAP) je komunikační protokol jehož účelem je výměna zpráv přes internetovou síť. Tyto zprávy jsou odesílány ve formátu XML pomocí HTTP protokolu. Velkou výhodou tohoto protokolu je nezávislost na programovacím jazyce právě díky formátu XML. V případě odesílání citlivých dat se doporučuje využití zabezpečeného protokolu HTTPS. SOAP je však flexibilní. Komunikace může také probíhat pomocí různých transportních protokolů jako je SMTP, TCP nebo například UDP.

SOAP je následníkem staršího protokolu XML-RPC. V případě protokolu SOAP je však k webové službě možné připojit WSDL dokument 2.6.0.1, díky kterému je využití webové služby snadnější [10].

Základní struktura SOAP zprávy je složena ze dvou základních částí. Jedná se o ”Header”a ”Body”, které jsou umístěny v hlavním elementu ”Envelope”(obálka celé zprávy). Hlavička (header) je nepovinný element a slouží především pro přenos dat určených například k identifikaci uživatele. Obsahuje tedy hlavně bezpečnostní prvky jakou jsou jméno a heslo klienta, využití digitálního podpisu případně šifrování zpráv a další.

SOAP však využívá dalších rozšíření jako je například WS-Security [11], které jak již název napovídá, rozšiřuje zprávu o bezpečnostní prvky. WS-Security je standard vydaný v roce 2004 organizací OASIS a obohacuje SOAP zprávu o další elementy definující například digitální podpisy, šifrování zprávy případně jen její části nebo také přenos časových razítek a tokenů. Elementy standardu WS-Security jsou umístěny v hlavičce zprávy a jsou obaleny elementem ”Security”.

Tělo zprávy (Body) pak přenáší data definující volanou metodu a její vstupní parametry nebo v případě odpovědi návratovou hodnotu.

2.6.0.1 WSDL [12] je dokument ve formátu XML, který využívá abstraktní definici například seznamu metod, které jsou webovou službou poskytovány. Taktéž tento dokument specifikuje přesný tvar jak požadavku, tak i odpovědi. Klient této služby má tak přehled nad tím, jaký vstup (jaké parametry) služba požaduje a jaké data se zpět vrátí. WSDL taktéž definuje datové typy jednotlivých parametrů. K definici datových typů WSDL využívá XML Schema Definition (XSD), díky kterému je možné definovat jak jednoduché datové typy (string, integer, boolean...) tak i složitější (complex types).

2.7 Další použité technologie

Mezi další využité technologie a nástroje, u kterých nebyl bližší popis podstatou, je WAMP webový server, nebo-li balíček obsahující Apache, MySQL a PHP pro operační systém Windows. Dále vývojové prostředí (IDE) Eclipse, Javascript a na něm postavenou knihovnu jQuery (UI).

3 Analýza platebních bran

Aktuální stav internetu je zahlcen systémy, které umožňují uživatelům provádět platby přes internet. Skutečností je, že v České republice stále převládá platba hotově při dobírce, což je způsobeno především nedůvěrou v internetové obchody, ale můžeme říci, že internetové platby začínají být postupem času oblíbenější a procento využití platby dobírkou při převzetí zboží klesá.

V této kapitole budou zmíněny nejpoužívanější platební systémy. Bude se jednat o 2 celosvětově známé systémy PayPal, Skrill a 4 české platební systémy PayU, GoPay, PaySec a GP webpay. Zmíněné platební systémy byly k bližšímu popisu vybrány především z důvodu jejich komplexnosti, případně počtu klientů, kteří tyto systémy využívají. Data uvedená u jednotlivých systémů byla získána především z oficiálních stránek těchto systémů ale také z poskytnutých údajů, o které bylo požádáno prostřednictvím elektronické pošty. Konkrétně platební systém PayPal je pak implementován právě v kreditním systému a právě z toho důvodu je tomuto systému věnováno více prostoru.

3.1 PayPal

Počátky vzniku platebního systému PayPal sahají do roku 1998. Založení PayPalu však nemělo vůbec jednoduchou cestu, což naznačuje i to, že oficiální vznik systému PayPal bylo až v srpnu 1999.

Zrod PayPalu stojí za dvěmi osobnostmi. Jsou jimi Peter Thiel a Max Levchin, kteří v září 1998 založili společnost FieldLink, zabývající se vývojem systému, pro zařízení Palm Pilots a ostaní PDA, kde systém uživatelům umožňoval využívat tato zařízení jako digitální peněženky. V prosinci 1998 byla pak společnost přeskupena a přejmenována na Confinity, jejichž hlavním cílem bylo umožnit převody peněz výhradně na PDA zařízeních. Jak už jsem zmínil, samotný PayPal, jako produkt, vznikl v srpnu 1999 pod společností Confinity. Ze začátku bylo jako bonus vloženo 10 dolarů na každý nově vytvořený PayPal účet. Tímto činem popularita PayPalu velice vzrostla a dostala se tak do podvědomí spousta lidí. V roce 2000 se pak společnost Confinity spojila se společností X.com, pod jménem X.com a v roce 2001 se pak tato společnost přejmenovala podle svého primárního projektu, PayPal. V říjnu 2002 byl pak PayPal odkoupen společností eBay za 1,5 miliardy dolarů. [13]

V současnosti se jedná o jeden z nejznámnějších a nejpoužívanějších platebních systém na světě, který je možné použít ve 190 zemích a 26 různých měnách zahrnující i českou korunu. V systému PayPal je zaregistrováno cca 143 milionů účtů, což svědčí o jeho oblíbenosti. Hlavním cílem je jednoduché, rychlé a bezpečné odesílání a přijímání internetových plateb případně převodů. Každý účet, jehož identifikace probíhá podle emailových adres, může být propojen s jednou nebo více platebními kartami. Po propojení PayPal účtu s kreditní kartou je uživatel ušetřen o opakované zadávání citlivých údajů kreditní karty. PayPal danou platbu zprostředkuje sám, zatímco uživatel zadá pouze e-mail a heslo. [14] [15]

Poznámka 3.1 Mezi podporované platební karty patří Visa, MasterCard, American Express, Discover, Maestro, Carte Aurore, Carte Bleue, Cofinoga, 4 étoiles, Carte Aura, Tarjeta Aurora, JCB.

3.1.1 Integrace PayPal

PayPal, přístupný jako webová služba, nabízí některé své funkce, jako možnost k integraci do webové aplikace klienta. Mimo jiné se jedná například o vytváření nového PayPal účtu s tím, že si uživatel může rovnou propojit nově vytvořený účet se svou kreditní kartou. Nejdůležitější možností je však integrace provádění plateb na straně klienta, k čemuž PayPal nabízí několik způsobů. Tyto způsoby jsou různé a liší se především rychlostí provedení platby. Mezi nejzajímavější a nejvíce používané patří především tyto:

- PayPal Payments Standard [16]
- Adaptive Payments [18]
- PayPal Payments PRO [17]
- Express Checkout [19]

První dvě zmíněné platební metody, PayPal Payments Standard a Adaptive Payments, budou především z důvodu jejich implementace v kreditním systému popsány podrobněji.

3.1.1.1 PayPal Payments Standard Tento způsob platby je asi nejpoužívanějším a to především z toho důvodu, že za něj není potřeba platit žádné měsíční poplatky narozdíl například od PayPal Payments PRO, kde aktuálně měsiční poplatek činí 30$. Tato způsob platby je postavena na základě HTML tlačítek různých typů. Tlačítka vždy odesílají HTML formulář, který obsahuje skryté položky definující dané tlačítko. Je v něm tedy definován typ tlačítka, částka k zaplacení, měna a další atributy nutné k provedení platby. K vytvoření tlačítka vede několik cest. Zmínil bych především vlastní sestavení HTML formuláře podle specifikace, vygenerováním přímo v profilu PayPal účtu nebo zavolání PayPal API metody BMCreateButton, která dokáže vrátit i zašifrované tlačítko, jehož formulářovým prvkům tak nelze podstrčit jiný obsah, než který byl definován systémem. Každé tlačítko se chová trochu jinak a jsou tak určeny pro různé typy plateb.

Buy Now Tlačítko umožňující provedení rychlé platby právě jednoho druhu zboží. Je však možné tlačítko upravit pomocí dodatečných možností jako je například umožnění výběru vlastnosti daného produktu (barva případně velikost oblečení atd.)

Donate tlačítko slouží převážně k odeslání nějaké darované částky. Tuto částku si plátce může buď zvolit sám nebo je částka již pevně stanovena.

Add to Cart tlačítko uživatelům webové aplikace umožňuje vybrat více druhů zboží, které se postupně vkládají do nákupního košíku s tím, že se nakonec zaplatí všechny vložené produkty najednou. "Add to Cart"tlačítko je spojené s "View cart"tlačítkem, které pak umožní zobrazení přehledu vybraného zboží

Subscribe slouží k prováděním pravidelných plateb, jako jsou například trvalé příkazy, platby za předplatné atd.

Buy Gift Certificate slouží k prodeji dárkových certifikátů přímo ve webové aplikaci

Nově vytvořené popřípadě vygenerováne tlačítko, si pak klient může vložit do své aplikace. Pomocí tlačíka jsou pak uživatelé webové aplikace přesměrováni na server platební brány PayPal, kde je požadováno přihlášení a v případě úspěšného zadání e-mailu a hesla je uživatel informován o detailu platby, kterou mu zbýva už jen potvrdit.

V kreditním systému bylo využito tlačítka BuyNow a to konkrétně k provedení vkladu prostředků.

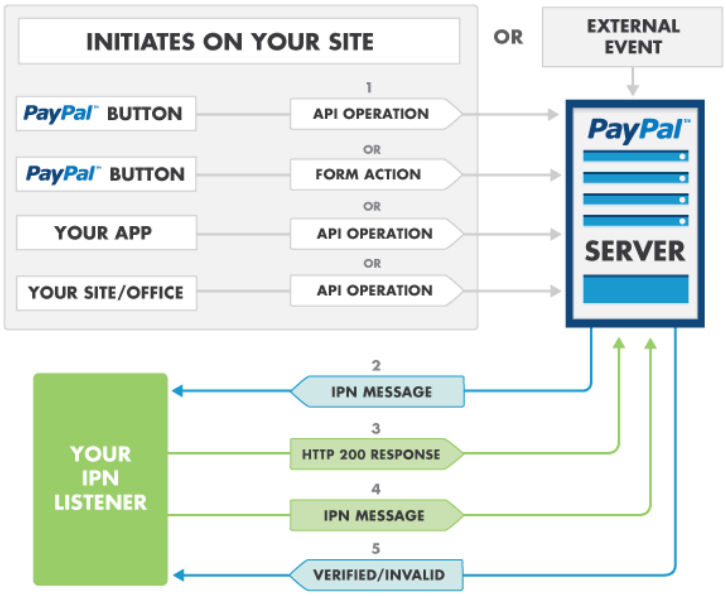
3.1.1.2 IPN notifikace Aby webová aplikace klienta získala informace o provedené transakci, tedy zda-li proběhla v pořádku nebo v jakém stavu se aktuálně nachází, PayPal má k dispozici tzv. instantní notifikace o platbě (IPN), díky kterým je klientská aplikace o aktuálním stavu transakce ihned informována a umožní tak klientovi evidenci provedeným transakcí na straně své aplikace.

Jedná se o HTTP POST požadavky, které PayPal odesílá na základě vytvořené platby. Odchytávání se provádí pomocí posluchače, kterého si klient naimplementuje sám podle dokumentace. Klientem vytvořený posluchač poté jen naslouchá na zvolené URL adrese a příchozí požadavky zpracovává. Může však dojít k tomu, že je posluchač buď špatně naimplementován nebo aktuálně není dostupný (výpadek serveru, rekonstrukce systému atd.). Z tohoto důvodu PayPal neodesílá požadavek s IPN zprávou jen jednou, ale hned několikrát v určitých intervalech. A to až do té doby, než ze strany klientské aplikace obdrží odpověď s HTTP kódem 200. Pokud počet pokusů odeslání dosáhne určité hranice, je toto odesílání zastaveno. Je však možnost odeslat zprávu ručně přímo z PayPal účtu. K umožnění testování posluchače pak PayPal nabízí nástroj, který bezplatně odesílá testovací notifikace přímo na webovou aplikaci klienta.

Takovýto posluchač je dobré využít i pro výše zmíněné tlačítka. Komunikace na základě IPN notifikací probíhající mezi webovou aplikací a platebním systémem PayPal je znázorněna na obrázku 1.

1. Odesláním tlačítka se vyvolá platba, která se dokončí v systému PayPal.
2. Po dokončení PayPal odešle HTTP metodou POST zprávu (notifikaci) o provedení platby IPN posluchači.
3. Posluchač vrátí prázdnou odpověď s HTTP kódem 200.
4. Posluchač pošle systému PayPal kompletní zprávu ve tvaru, ve kterém přišla (obsah zprávy se nesmí změnit).

5. PayPal pak už jen pošle rozhodnutí, zda-li proběhlo ověření v pořádku, či nikoliv. Jedná se o zprávy "VERIFIED"případně "INVALID".



Obrázek 1: Tok IPN notifikací [20]

Těchto 5 kroků zahrnuje ověření (konkrétně kroky 2-4), která klienta ujistí o tom, že IPN notifikace přišla právě ze strany systému PayPal, což je potvrzeno již výše zmíněnými odpověďmi "VERIFIED"nebo "INVALID".

IPN notifikace obsahuje několik proměnných (atributů), které nesou informace o transakci. Obsahuje tedy například email plátce a příjemce, celkovou sumu, poplatek, stav a další informace o provedené transakci.

3.1.1.3 Adaptive Payments [21]

Jedná se o rozhraní API, které umožňuje provádění jak jednodužších, tak složitějších platebních operací. Adaptive payment API obsahuje metodu "Pay", která provádí platby mezi plátcem a jedním nebo více příjemci v závislosti na tom, jaký typ platby byl vybrán. K dispozici jsou 3 typy plateb:

Simple je typ platby, kde vystupuje právě jeden plátce a jeden příjemce.

Parallel umožňuje plátci odeslat platbu více příjemcům.

Chained nebo-li zřetězená platba, umožňuje odeslat platbu na jednoho primárního příjemce, který pak zbytek může rozeslat na příjemce sekundární.

Proces schvalování plateb probíhá na straně plátce a způsobů, jakými je možné platbu schválit, je hned několik.

Explicitní Plátce je přesměrován do rozhraní systému PayPal, kde následně platbu schválí.

Implicitní Aplikace je zároveň i jako plátce a platba je tak schválena implicitně.

Dopředné schválení Uživatel si ve svém PayPal účtu může navolit platby, které určí jako předem schválené. Další schválení tak už není potřeba.

Vložení do aplikace Proces schválení je vložen přímo do aplikace. Uživatel není tedy přesměrován do systému PayPal. Výhodou tohoto způsobu je možnost přeskočení kroku přihlášení, díky čemuž je proces schválení urychlen. PayPal si plátce zapamatuje uložením do cookie jeho prohlížeče. Jedná se tedy o méně bezpečnou variantu, kdy je možné tohoto zapamatování zneužít.

V kreditním systému je zprovozněn typ platby "Simple"v kombinaci se dvěmi způsoby schválení. Jde o schválení implicitní a explicitní přičemž aktivní může být vždy jen jeden způsob v závislosti na tom, jak je systém aktuálně nastaven. Kromě metody "Pay"byla z AdaptivePayments API v kreditním systému využita také metoda "PaymentsDetails"díky které bylo možné získat bližší informace o platbě.

3.1.1.4 Výhody PayPal

- Rozšíření po světě.
- Bezpečnost.
- Žádný poplatek za založení účtu.
- Žádné měsíční poplatky.
- Podpora při integraci.
- Testovací prostředí pro integraci (sandbox).

3.1.1.5 Neýhody PayPal

- Transakční poplatky.
- Vysoký poplatek za špatné zadání čísla účtu.
- Poplatky za každou transakci.

3.2 Skrill (MoneyBookers)

Počátky systému Skrill sahají do roku 2001. V současné době je v této společnosti zaměstnáno 560 zaměstnanců. Skrill, v minulosti známý jako MoneyBookers, je další mezinárodní platební systém nebo-li internetová peněženka, která umožňuje podobně jako PayPal posílat nebo přijímat peníze po celém světě. Vše co k používání Skrillu uživatel potřebuje, je vlastní e-mailová adresa, pomocí které se provádí veškeré platby. Skrill umožňuje zaregistrovat pouze jeden účet, na který si však uživatel může vložit až 5 e-mailových adres, pod kterými se pak může autentifikovat. V současnosti vlastní Skrill účet přes 36 milionů zákazníků a 156 000 obchodníků. [22]

Skrill umožňuje 2 způsoby, jak si peníze vložit na Skrill účet.

Bankovní převod Tento způsob vkladu je asi nejpoužívanější. Hlavní důvod je ten, že se jedná o bezplatný způsob, jehož menší nevýhodou je doba trvání, pohybující se mezi 2 až 5 dny. Po výběru této metody, Skrill vygeneruje veškeré údaje potřebné k provedení platby.

Kreditní kartou Způsob vkladu pomocí kreditní karty je zpoplatněn 1.9 % z vložené částky, avšak provedení vkladu proběhne ihned. Skrill podporuje v České republice kreditní karty typu Visa, MasterCard, American Express, JCB případně Dinners Club a debetní karty typu Visa Electron.

Kromě vkladu přes bankovní účet jsou bezplatné taktéž všechny příchozí transakce. Odesílání peněz je pak zpoplatněno 1% odeslané částky s tím, že aktuálně poplatek nepřekročí částku 10EUR.

Výhody Skrill

- Možnost odeslání peněz na e-mail jiných uživatelů.
- Bezplatné založení účtu.
- Bezpečnost.
- Podpora češtiny.

Neýhody Skrill

- Vysoký poplatek při výběru peněz, který se pohybuje mezi 2.95 - 3.50 EUR.

3.3 PayU

Český platební systém podporující širokou škálu platebních metod jako je například Mojeplatba od Komerční banky, Platba 24 od České spořitelny, mPeníze od mBank, ePlatby od Raiffeisen Bank, GE Money Bank a další, které nabízí takzvaný rychlý online převod. PayU taktéže podporuje platební karty VISA, MasterCard a DinnersClub [23]. Výše poplatků za zpracování on-line plateb nemá tento systém pevně stanoven. Výše poplatku je určena podle průměrného obratu, průměrné výše transakce on-line plateb a také typu

obchodní činnosti uživatele [24]. Veškeré pohyby peněz jsou zabezpečeny protokolem SSL a 3D Secure řešením.

PayU nabízí komplexní řešení a plnou podporu při integraci do klienteského systému. Takových systémů je na ůzemí České Republiky cca 1000, mezi které patří například největší český obchodní portál Aukro.

3.3.1 Proces provedení platby

Proces platby je jednoduchý, rychlý a bezpečný, což je hlavní motto PayU. V případě, že zákazník provádí platbu za zboží či službu pomocí rychlé online platby, má nejdříve na výběr jednu z výše uvedených podporujících bank. Po výběru své banky je zákazník přesměrován do jeho internetového bankovnictví, kde se přihlásí. V případě úspěšného přihlášení se zákazníkovi zobrazí již předvyplněný platební příkaz, který musí už jen potvrdit, čímž ja platba dokončena.

V případě, že zákazník upřednostňuje platbu kartou, jednoduše zvolí tento typ platby (Visa, MasterCard). Zákazník je následně přesměrován na stránku s formulářem, kde je vyzván k zadání osobních údají spolu s údaji o kreditní kartě, mezi které patří číslo karty, datum expirace a CVV kód.

3.4 GoPay

GoPay je stejně jako PayU český produkt, jehož počátky sahají do roku 2007. Provoz platebního systému GoPay byl však spuštěn o rok později. V roce 2010 se tento platební systém stal prvním systémem v ČR, který pro platby využíval platební metody třetích stran. Postupem času se GoPay rozšířil i na Slovensko, což také pomohlo k tomu, že v roce 2013 využívalo GoPay již 3000 obchodních míst. [25]

Platební systém GoPay nabízí možnost vytvoření elektronického účtu, nebo-li elektronické peněženky běžným uživatelům. Pro obchodní subjekty jako jsou například internetové obchody pak GoPay vyjde vstříc se svou platební branou. K vytvoření této brány je však potřeba nejdříve vlastit GoPay účet, který je považován za základní a centrální produkt platebního systému GoPay. Na tento účet jsou uživatelům připisovány veškeré platby. Bez vlastního GoPay účtu by nebylo možné využívat dalších služeb jako je například již zmíněná platební brána.

API pro integraci platební brány GoPay je podporována v 6 základních programovacích jazycí. Jedná se o ASP.NET, ASP Classic, Java, PHP, Python a Ruby. K integraci platební brány pak slouží rozsáhlá a plnohodnotná dokumentace.

3.4.1 Proces provedení platby

Proces provedení platby u platebního systému GoPay je pak obdobný procesu u výše zmíněného platebního systému PayU. Uživatel si na e-shopu, jenž má tuto platební bránu implementovanou, vybere způsob platby který, vyžaduje. V závislosti na tom, zda-li se jedná o český nebo slovenský trh, jsou nabízeny různé platební metody. Jak u české, tak u slovenského trhu je k dispozici platba kartou, online bankovním tlačítkem, bankovním

převodem, mobilem a nebo pomocí elektronické peněženky (GoPay). U českého trhu je však navíc možnost provedení kupónových plateb.

Po výběru požadované metody je uživatel přesměrován do platební brány GoPay, kde je platba dokončena.

3.5 PaySec

PaySec je další z českých platebních bran, která je však součástí finanční skupiny ČSOB. PaySec umožňuje provádění mikroplateb v internetových obchodech, ale taktéž vychází vstříc neziskovým organizacím svou integrovanou bránou "Klikni a daruj" [26].

Mezi platby, které PaySec podporuje patří platby kartami MasterCard, VISA a Diners Club. Platby je taktéž možné provádět online s elektronického bankovnictví ČSOB a Poštovní spořitelny případně Ery. Jednotlivé platební metody je pak možné integrovat několika způsoby. Plnou integrací se do internetového obchodu integrují všechny metody. Další možností je integrace darovacího tlačítka v případě neziskových organizací dále pak například provedení převodu z rozhraní PaySec nebo provedení výzvy k platbě z konta obchodníka.

Při započítání držitelů platebních karet, uživatelů internetového bankovnictví ČSOB a Era, uživatelů aplikace MasterCard Mobile, uživatelů elektronické peněženky PaySec se celkový počet klientů brány PaySec pohybuje v jednotkách milionů.

Tato platební brána, která jako svou největší konkurenci uvádí již zmíněné systémy GoPay nebo PayU, zařazuje mezi své nejvýznamější klienty mimo jiné největší obchod s počítačovou technikou Alza dále pak České dráhy, ČSA nebo například Student Agency.

3.6 GP webpay

GP webpay je jedna z nejrozšířejnějších platebních bran v České republice, kterou využívá více než 2000 e-shopů. GP webpay umožnuje platby asociací MasterCard, VISA a Diners Club a to jak tuzemskými, tak zahraničními kartami. Mezi partnery pak patří banky ČSOB, KB, Raiffeisen Bank a nebo UniCredit Bank. Výše poplatků je pak účtována individuálně v závislosti na dohodě s bankou. [27]

4 Softwarová laboratoř

Vytvoření softwarové laboratoře a dále její rozvoj bylo základem k tomu, aby bylo možné pracovat na projektech zahrnující konkrétní témata diplomových prací. Dlouhodobým cílem této laboratoře je vytvořit a rozvíjet prostředí pro výuku, výzkum a vývoj softwarového inženýrství.

4.1 Založení softwarové laboratoře

Na vytvoření softwarové laboratoře se podílelo 8 členů pod vedením 3 mentorů, kteří rovněž zastávali funkci zadavatelů. V první fázi vytváření softwarové laboratoře bylo potřeba rozdělit členy do několika týmů. Vzhledem k tomu, že se jednalo o 2 projekty, rozdělili jsme se do 2 týmů. S ohledem na složitost projektů, byli členové do týmů zařazeni v poměru 3:5. První tým byl tedy složen ze tří členů a věnoval se vývoji mobilních aplikací, druhý tým byl pak složen z pěti členů a pracoval na vývoji komponent pro webové aplikace a také samotnou webovou aplikaci s tím, že se vytvořené komponenty ve webové aplikaci využíjí, čímž se především otestuje jejich funkčnost.

Já jsem byl osobně přiřazen ke druhému týmu, tedy k vývoji webové aplikace a komponent, jemuž bych v této sekci věnoval pozornost.

V rámci našeho týmu bylo cílem každého člena (mimo team leadera) naimplementovat svou vlastní komponentu, dostupnou jako webovou službu a dále spolupracovat na systému, na kterém budou všechny komponnety použity.

Ve druhé fázi vytváření softwarové laboratoře se v rámci týmu rozdělily role, jakých budou jednotliví členové zastávat. A protože bylo potřeba více lidí pro vývoj, byl zvolen jeden team leader, který tým vedl metodikou SCRUM a 4 vývojáři, kde každý vyvíjel vlastní komponentu.

V další fázi bylo potřeba sjednotit tým a vytvořit společné pracovní a vývojové prostředí. K dispozici jsme dostali přístupy ke školnímu serveru, což nám umožnilo zveřejnit naše aplikace a komponenty a prezentovat je tak našim zadavatelům. Dále bylo zapotřebí zavést nějaký systém k vytváření úkolů a jejich přiřazování k jednotlivým členům týmu. K tomu jsem využili webový systém Mantis. Díky tomuto systému jsme měli větší přehled nad všemi úkoly nebo například i nad tím v jakém stavu se aktuálně nacházejí a kdo na nich pracuje. Mantis byl taktéž zprovozněn na školním serveru, díky čemuž měli všichni členové týmu, včetně mentorů, přístup k jednotné instanci tohoto webového systému. Dále bylo potřeba vývoj nějakým způsobem verzovat a zpřístupnit všem členům týmu aktuální stav aplikace, čímž mám na myslim především zdrojové kódy, a mimo jiné zamezit možnost přepisování kódů mezi členy týmu. K tomu jsme na našem serveru na základě verzovacího systému GIT vytvořili repositář pro naše projekty. Díky Gitu a zpřístupněného serveru bylo pak možné zveřejnit určitou verzi systému. Díky tomuto způsobu nasazení aplikace na server bylo předejito častým problémům, které vznikají při ručním kopírování aplikace například pomocí FTP klienta, což bývá především časově náročné. Námi vytvořený repozitář měl striktně danou adresářovou strukturu. Bylo určeno, kde se nachází veškeré dokumenty k projektům, migrace databází a samotné aplikace.

Jakmile bylo vše potřebné na serveru nainstalováno a zprovozněno, zvolili jsme si jednotné vývojové prostředí (IDE). Jednalo se o Eclipse, mimo jiné i z důvodu možnosti doinstalování potřebných pluginů, které v našem případě byly potřeba. Jednalo se například o plugin umožňující vývoj v jazyce PHP (PDT - PHP development tool), ve kterém se naše aplikace vyvíjely, nebo také plugin pro zjednodušení práce s verzovacím systémem GIT, kde Eclipse nabízí plugin eGIT. Také bylo potřeba sjednotit server k umožnění vývoje na lokálních pracovních stanicích všech členů týmu. Byl zvolen balík WAMP, který obsahuje potřebné nátroje pro vývoj webových aplikací (Apache, PHP a MySQL).

Tímto bylo vytvořeno základní pracovní prostředí pro vývoj aplikací našeho typu, které se bude nadále zdokonalovat, čímž se vývoj stávajících nebo i nových systémů může významně urychlit.

5 Kreditní systém plateb

V této kapitole se budu věnovat primárnímu cíli mé diplomové práce, čímž bylo vytvoření komponenty pro kreditní systém plateb na webu. Kromě popisu a objasnění funkcionality kreditního systému bude tato kapitola také obsahovat specifikaci požadavků, analýzu, návrh a také implementaci.

5.1 Popis systému

Kreditní systém plateb umožňuje uživatelům provádět platební transakce ve více webových aplikacích pod jedním uživatelským účtem a mít tak kompletní přehled na jediném místě. Jedná se také o webovou službu, se kterou můžou komunikovat, pomocí komunikačního protokolu SOAP, jednotlivé webové aplikace (klientské aplikace) požadující platby za zboží nebo služby. Tato webová služba může ušetřit spoustu práce a času při vývoji klientských aplikací, kde je jediným úkolem v případě kreditního systému integrace plateb podle toho, jaký typ plateb (viz kapitola 5.2) je pro danou aplikaci vyhovující. Webová služba také nabízí možnosti, jako je vyžádání virtuálního účtu webového uživatele přiřazeného dané aplikaci nebo zjištění zůstatku na tomto účtu případně kreditů, které jsou blokovány. Seznam metod, poskytovaných webovou službou se však bude nadále rozšiřovat.

Webová služba je přístupná jen těm webovým aplikacím, které si kreditní systém zaregistrují, čímž získají svůj unikátní identifikátor a vygenerované heslo, které si mohou vyzvednout přímo ve svém účtu. Tato registrace je umožněna v centrálním systému (viz kapitola 6), na který je kreditní systém plateb napojen.

Kreditní systém je také napojen na uživatelský systém na kterém je závislý. Účelem uživatelského systému je kompletní správa uživatelů, jejich autentizace i autorizace. Právě díky tomuto systému, který je taktéž produktem softwarové laboratoře, je umožněno uživatelům provádět platby ve více webových aplikacích pod jedním uživatelským účtem.

Kromě toho, že kreditní systém plateb slouží jako webová služba, má také své vlastní uživatelské prostředí, ke kterému mají přístup weboví uživatelé. Uživatelské prostředí kreditního systému plateb nabízí především kompletní správu virtuálních účtů, tedy vytváření, mazání nebo přepínání mezi jednotlivými webovými aplikacemi, dále přehled o stavu svých kreditů (jaký je zůstatek popřípadě kolik kreditů je blokovaných) s možností libovolného převádění mezi virtuálními účty a nebo kompletní přehled o své transakční historii. K tomu, aby bylo webovým uživatelům umožněno kredity dobíjet popřípadě vybírat, bude kreditní systém dále napojen na bránu PayPal. Platební systém je navrhnut tak, aby bylo umožněno přidat libovolný počet platebních bran, s čímž se do budoucna počítá. PayPal byl prozatím vybrán jako výchozí, i z toho důvodu, že námi požadované operace poskytuje bezplatně, nabízí kvalitní testovací rozhraní a až na některé vyjímky nevyžaduje jakoukoliv komunikaci týkající se například nastavení brány.

Poznámka 5.1 Virtuální účty plní podobnou funkci jako klasické bankovní účty. Slouží tedy k uchovávání kreditů s tím, že počet virtuálních účtů není jakkoliv omezen. Tyto

účty se pak přiřazují jednotlivým klientským aplikacím. Virtuální účet je reprezentován 9-ti místnou kombinací čísel, kterou si užival může buď nechat vygenerovat a nebo si zvolí svou vlastní.

5.2 Typy transakcí

Kreditní systém poskytuje několik typů transakcí. Jedná se o transakce, které jsou proveditelné buď v uživatelském prostředí kreditního systému, bavíme se o transakcích typu vklad, výběr nebo převod mezi účty, a nebo přímo v klientské aplikaci, kde se jedná o konkrétní platby za položky dané aplikace. Tyto platby v klientských aplikací budou dvou typů. V první řadě se bude jednat o platbu aplikaci, v druhé řadě pak o platbu jinému uživateli, kde klientská aplikace bude zastávat roli zprostředkovatele. Druhý zmíněný typ však prozatím není zprovozněn.

Ke všem výše zmíněným typům transakcí je možno nastavit konkrétní poplatky. 5.3

Platba v klientské aplikaci (Uživatel-uživatel) Jedná se o způsob platby, kde uživatel webové aplikace platí jinému uživateli. Mezi typy webových aplikací, kde by mohl být tento typ transakce užitečný, by mohly patřit například bazary. Správce aplikace si však může, ale nemusí, jakožto zprostředkovatel služby, zvolit procento, které se mu z platby bude odvádět na jeho vlastní virtuální účet, který mu je při registraci kreditního systému vygenerován. Tento typ transakce, jako jediný není prozatím v systému zprovozněn. S rozšířením o platbu mezi uživateli se však dopředu počíta.

Platba v klientské aplikaci (Uživatel-aplikace) V tomto případě se jedná o platbu v klientské aplikaci, kterou využijí především internetové obchody. Jedná se o platbu, kde příjemcem platby je právě klientská aplikace. Kredity jsou tedy převedeny z virtuálního účtu uživatele na virtuální účet vygenerovaný pro danou aplikaci.

Vklad prostředků Vklad prostředků je prozatím proveden pomocí platebního systému PayPal. Kreditní systém je však připraven tak, že je možné systém rozšířit o jiné platební systému. Pomocí vkladu, jsou uživateli přičteny kredity na jeho primární virtuální účet, se kterými může provádět platby v aplikacích zaregistrovaných v kreditním systému.

Výběr prostředků Stejně jako transakce typu vklad, je i tento typ transakce zprovozněn pomocí platebního systému PayPal. Uživatelé, kteří své účty využívají za účelem výdělku tedy potřebují alespoň jeden vlastní PayPal účet, na který budou kredity vybírány.

Převod mezi uživateli Užitečný typ transakce, díky kterému mohou uživatelé kreditního systému převádět kredity mezi svými účty. Účet jiného uživatele může být specifikován buď uvedením konkrétního účtu a nebo e-mailu uživatele, přičemž u druhého zmíněného způsobu jsou kredity zaslány na primární účet uživatele.

5.3 Poplatky

Jedná se o další typ transakce. Je však specifickým typem, který bude v této kapitole podrobněji popsán.

Specifický je především v tom, že se dynamicky vyhodnocuje a uplatňuje podle zvoleného typu transakce. Každý typ transakce může mít neomezené množství poplatků. V praxi se však počítá s využitím jednoho nebo v ojedinělých případech maximálně dvou poplatků v jednu chvíli. Nicméně poplatky je možno definovat tak, že budou odlišné při různé výši platby viz níže. Jde zde především o jejich administrovatelnost s možností přidělení k jednotlivým typům transakcí.

Při zavádění poplatku do systému je možná volba ze dvou typů poplatku. Jde o procentuální a absolutní poplatek. U procentuálního poplatku je jeho výše stanovana výpočtem z celkové částky platby. Absolutní poplatek je pak pevný a nezáleží tak na výši platby.

U poplatku je taktéž možno definovat, za jakých okolností bude uplatněn:

1. Stanovení horní hranice (Poplatek je uplatně pouze v případě, že částka nepřesahuje tuto hranici)
2. Stanovení dolní hranice (K uplatnění poplatku dochází v případě, že částka tuto hranici přesahuje)
3. Stanovení dolní i horní hranice platby (V případě, že se částka k zaplacení pohybuje v definovaném rozmezí, je poplatek uplatněn)

Na stejném principu pak funguje uplatnění poplatků v definovaném časovém období.

5.4 Specifikace požadavků

Jedná se o první fázi vývoje, jejimž cílem bylo zachytit všechny požadavky zadavatele. Budou zde tedy popsány jednotlivé případy užití, které budou znázorněny v diagramu případu užití, kdo se s tímto systémem dostane do styku a jaké bude mít daný aktér pravomoce. Pomocí scénářů poté podrobněji popíši jednotlivé případy užití (funkce systému).

V následujících bodech budou shrnuty zadané požadavky:

- Kreditní systém bude samostatná komponenta frameworku Symfony ve formě bundlu (viz 2.3).
- Kreditní systém bude pomocí definovaného rozhraní využívat uživatelský systém jako správce uživatelských účtů, ale také jako autorizační a autentifikační službu.
- Veškerá správa kreditního systému bude probíhat v centrálním systému.
- Kreditní systém bude přístupný jako SOAP webová služba pro klientské aplikace, která bude umožňovat:
  - Provádění plateb

  - Získání informace o stavu na účtu (zůstatek a blokované kredity).

- Webový uživatel kreditního systému bude mít možnost nad virtuálními účty provádět následující operace:
  - Vytváření libovolného množství účtů v různých měnách.
  - Aktivování a deaktivování jednotlivých účtů.
  - Možnost volby primárního účtu.
  - Jednotlivé virtuální účty bude možné přiřazovat k jednotlivým(zaregistrovaným) aplikacím.
- Uživatel bude mít přehled nad stavy svých účtů (aktuální zůstatek a blokované kredity).
- V základním stavu kreditního systému bude možné kredity dobíjet pomocí platební brány PayPal, taktéž výběr bude prováděn touto platební branou.
- Uživatel bude mít přehled nad svými virtuálními i PayPal transakcemi.
- Kreditní systém umožní převody mezi jednotlivým virtuálními účtu a to buď určením konkrétního virtuálního účtu a nebo e-mailu příjemce.

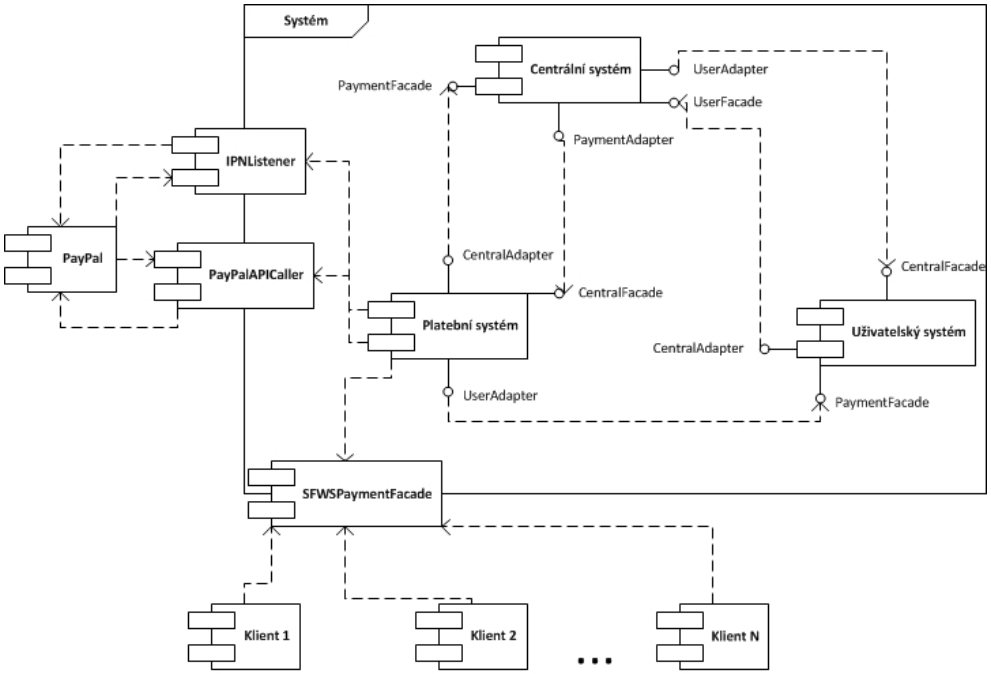
5.4.1 Architektura systému

Kompletní systém se skládá z několika komponent, jejichž počet se může do budoucna rozšiřovat podle potřeby. Komponenty je možné do systému přidávat nebo ze systému odebírat včetně komponent, na ni závislých. Aktuálně je systém složen ze 3 hlavních komponent, které obsluhují svou vlastní databázi.

- Centrální systém (viz. 6)
- Kreditní systém
- Uživatelský systém

Každá z uvedených komponent má však pevně stanoveno, na kterou komponentu je závislá, nebo-li kterou potřebuje ke své funkcionalitě. Pro veškerou komunikaci jsou v jednotlivých komponentách předdefinována rozhraní jak na straně ”požadované”komponenty (komponenta potřebná pro správnou funkci jiné komponenty), tak na straně komponenty ”požadující”(komponenta požadující funkcionalitu jiné komponenty) . Jedná se o rozhraní, sloužící jako přístupové body k jednotlivým komponentám případně jako body výchozí.

Přístupový bod je vždy definován jako návrhový vzor ”fasáda”, zatímco u výchozího bodu se jedná o ”Adaptér”. Díky těmto rozhraním je možné oddělit od zbytku aplikace funkcionalitu, vyžadující jinou komponentou nebo v případě adaptéru definovat, pomocí kterých (vlastních) metod přistupovat k metodám jiné komponenty a adaptovat je tak na své vlastní.



Obrázek 2: Diagram komponent

Z diagramu 2 lze tedy vyčíst, že kreditní systém je závislý na obou komponentách, jak na centrálním, tak i na uživatelském systému. Zátímco mezi komponenty, které kreditní systém využívají, patří pouze centrální systém. Centrální systém, který blíže popíši v kapitole 6, je jak již název napovídá centrálním uzlem všech komponent, kde bude k dispozici jejich správa. Mezi jeho závislosti budou tedy spadat všechny komponenty, které bude systém obsahovat.

Součástí kreditního systému jsou taktéž třídy IPNListener, PayPalAPICaller nebo SFWSPaymentFacade. Jedná se o třídy, které jakýmkoliv způsobem komunikují s okolním světem. Právě z tohoto důvodu a z důvodu lepšího znázornění jsou v diagramu komponent vyčleněny z kreditního systému.

IPNListener Jedná se o posluchač, který očekává IPN zprávy z PayPal systému a nadále je zpracovává. Jedná se o oboustranou komunikaci viz. 1

PayPalAPICaller PayPal má k dispozici širokou škálu metod určených například pro provedení plateb. Kreditní systém některé z těchto metod využívá a právě třída PayPalAPICaller je za volání těchto metod a následné zpracování zodpovědná.

SFWSPaymentFacade Jak je již možné z názvu vyčíst, jedná se o další fasádu (návrhový vzor), která pro svůj účel webové služby využívá potřebných tříd kreditního systému a slučuje tak metody, které bude webová služba poskytovat, právě do jedné konkrétní třídy. Provedení plateb a jiné operace jako například zjištění zůstatku nebo

virtuálního účtu jsou pak k dispozici všem klientským aplikacím pomocí právě této webové služby. Kompletní výpis metod poskytovaných webovou službou, popisuje kapitola 5.7.1.

5.4.2 Diagram případu užití

Diagram případu užití (Obrázek 3), nebo-li UseCase diagram je grafický náhled na systém, díky kterému je jednoduše rozpoznatelné, kdo se systémem bude pracovat a jakých funkcí může využívat (případy užití).

Z diagramu případů užití 3 je patrné, že ke kreditnímu systému mohou přistupovat uživatelé různých rolí. Jedná se o role super administrátor, dále jen superadmin, administrátor, dále jen admin, a webový uživatel. Každá tato role má definovány pravomoce k provedení určitých operací.

5.4.3 Webový uživatel

Rolí webového uživatele je označen uživatel klientské aplikace, která využívá kreditní systém. Každý zaregistrovaný uživatel tak má možnost přihlášení do kreditního systému, kde má přehled o svých transakcích provedených v jednotlivých aplikacích, možnost převádení kreditů, výběru, vkladu a taktéž má k dispozic kompletní správu svých virtuálních účtů, které si může k jednotlivým zaregistrovaným aplikacím přiřazovat.

5.4.4 Admin

Administrátor má přístup do centrálního systému, odkud je platební systém řízen. Prozatím je administrátorovi umožněna registrace vlastních aplikací, registrace komponent k těmto aplikacím a samozřejmě správa komponent, mezi které patří právě i kreditní systém. V rámci kreditního systému pak má admin k dispozici přehled nad všemi transakcemi svých aplikací a taktéž má možnost k těmto aplikací vytvářet a přiřazovat poplatky. Při zaregistrování kreditního systému k určité aplikaci je vygenerován virtuální účet, u kterého má admin přehled o zůstatku a počtu blokovaných kreditů.

5.4.5 Superadmin

Stejně jako admin, tak i superadmin přistupuje přímo do centrálního systému, odkud kreditní systém, ale i ostatní komponenty řídí. Superadmin má však přehled nad všemi transakcemi a může vytvářet a spravovat jak globální poplatky (poplatky vztahující se na všechny aplikace), tak poplatky konkrétním aplikacím. Superadmin má taktéž vygenerován virtuální účet, který je prozatím určen k příjmu poplatků.

Z důvodu rozsáhlosti scénářů bude v této kapitole uveden pouze jeden a to scénář popisující případ užití "Paypal vklad"nebo-li proces k vložení finančních prostředků do kreditního systému. Ostatní scénáře jsou pak uvedeny mezi přílohami viz příloha D.

Obrázek 3: Diagram případů užití

- **Případ užití:** PayPal vklad
- **Předpoklady:**
  1. Aktér je přihlášen do kreditního systému.
  2. Aktér má vytvořen primární virtuální účet.
  3. Superadmin má nastaven PayPal účet.
- **Hlavní scénář:**
  1. Aktér stiskne tlačítko „Vklad“ .
  2. Systém přesměruje aktéra do rozhraní PayPal.
  3. Aktér zvolí výši požadovaného vkladu.
  4. POKUD aktér nemá vlastní PayPal účet, pokračuje se alternativním scénářem.
  5. POKUD aktér vlastní PayPal účet.
     (a) Aktér vyplní přihlašovací údaje v záložce „Pay with my PayPal account“.
     (b) Aktér pokračuje kliknutím na tlačítko „Log In“.
     (c) Aktér je přesměrován do dalšího kroku platby, kde si může zvolit adresu a platbu.
     (d) Aktér pokračuje kliknutím na tlačítko „Pay Now“.
     (e) PayPal provede transakci a odešle IPN zprávu do kreditního systému.
     (f) Kreditní systém příjme IPN zprávu a z důvodu validace ji odešle zpět na PayPal.
     (g) Paypal ověří IPN zprávu a potvrzení odešle kreditnímu systému.
     (h) POKUD je IPN zpráva validní a zakódované ID uživatele je korektní.
        i. Kreditní systém zavede paypal účet plátce do systému pokud zjistí, že dosud evidován nebyl.
        ii. Kreditní systém zaeviduje paypal transakci včetně poplatku, který je evidován jako další transakce.
        iii. POKUD je paypal transakce ve stavu Dokončeno.
           A. Kreditní systém zjistí primární virtuální účet plátce.
           B. POKUD je primární virtuální účet v jiné měně než ve které byla provedena PayPal transakce, bude částka převedena podle aktuálního kurzu České národní banky.
           C. Kreditní systém vytvoří novou transakci určující vklad na primární virtuální účet včetně poplatků vstahujících se k tomuto typu transakce.
  6. Proces vkladu pomocí brány PayPal byl dokončen.
- **Alternativní scénář**:
  1. Aktér vyplní formulář v záložce „Pay with a debit or credit card“.
  2. Aktér pokračuje kliknutím na tlačítko „Pay“.
  3. Proces vkladu pokračuje bodem 4.e hlavního scénáře.

5.5 Analýza a návrh

5.5.1 Datová analýza

Výsledkem datové analýzy, kterou jsem využil pro popis struktury databáze a tím tak databázi zpřehlednil a ulehčil její tvorbu ve fázi implementační, je lineární zápis typů entit a jejich atributů a E-R diagram.

5.5.1.1 Lineární zápis typů entit

- users (id, user_id, type)
- notifications (id, message, created, read, *user_id*)
- paypal_accounts (id, email, street, postcode, *account_id*)
- virtual_accounts (id, account_number, main, *account_id*)
- accounts (id, active, created, deleted, *user_id*, *currency_id*)
- accounts_applications (id, added_date, removed_date, *id_application*, *id_account*)
- currencies (id, shortcut, htmlcode, decimal, hidden)
- paypal_transactions (id, tax, firstname, lastname, street, city, state, country, postcode, telephone, email, paypal_transaction_id, paypal_parent_transaction_id, paypal_data, pay_key, *transaction_id*)
- virtual_transactions (id, custom_id, custom_description, *transaction_id*)
- transactions (id, sender_amount, receiver_amount, created, confirmed, rate, *state_id*, *transaction_type_id*, *receiver_account_id*, *sender_account_id*, *transaction_id*, *application_id*)
- applications (id, appliaction_id, currency_id, *transaction_type_id*)
- transaction_types (id, name, code, is_client)
- states (id, name, code, paypal)
- fee_types (id, code, name)
- fees (id, amount, description, from_amount, to_amount, from_date, to_date, active, *fee_type_id*, *transaction_type_id*, *account_id*, *application_id*)

5.5.1.2 E-R Diagram znázorňuje databázovou strukturu aplikace. Mezi nejdůležitější databázové tabulky kreditního systému, jehož struktura databáze je znázorněna v "Entity Relationship Diagramu"na obrázku 4, bych zařadil především tabulky transactions a accounts. Jedná se o tabulky, na kterých je celý systém postaven. Záměrem celého systému byla mimo jiné také možnost rozšíření o další platební brány. Aktuálně jsou k dispozici PayPal transakce, které definují transakce provedené pomocí systému PayPal a virtuální transakce provedené samotným kreditním systémem. Struktura databáze tedy byla navržena tak, že je vytvořena jedna obecná tabulka, transactions, společná prozatím tedy pro PayPal a virtuální transakce. Obecná tabulka "transactions"je následně rozšířena o tabulku "paypal_transactions"případně "virtual_transactions"v závislosti na tom, o jaký typ transakce se jedná. Tyto rozšiřující tabulky pak obsahují specifická data pro každý typ transakce. V případě implementace nového platebního systému, pomocí kterého budou mít uživatelé možnost provádět vklad a výběr prostředků, je tedy především potřeba vytvořit nové rozšíření tabulky transactions, které bude pak obsahovat svá specifická data.

Na stejném principu je pak založena tabulka accounts, která je rozšířena tabulkami "paypal_accounts"a "virtual_accounts", obsahující atributy specifické danému účtu.

Z E-R diagramu bych dále vyzdvihl tabulky "applications"a "users". Jedná se o tabulky, které vytváří tzv. můstek mezi kreditním systémem a s nim komunikujícími systémy. Mezi tyto systémy patří uživatelský a centrální systém. Každý systém má svou vlastní databázi. Pomocí můstků je pak možné namapovat jednoznačné identifikátory tabulek jiných databází na své vlastní. V případě uživatelského systému si kreditní systém pomocí můstku namapuje jakéhokoliv uživatele. V případě centrálního systému je pak v kreditním systému potřeba namapovat aplikace, které jsou dále využity při přiřazování k virtuálním účtům.

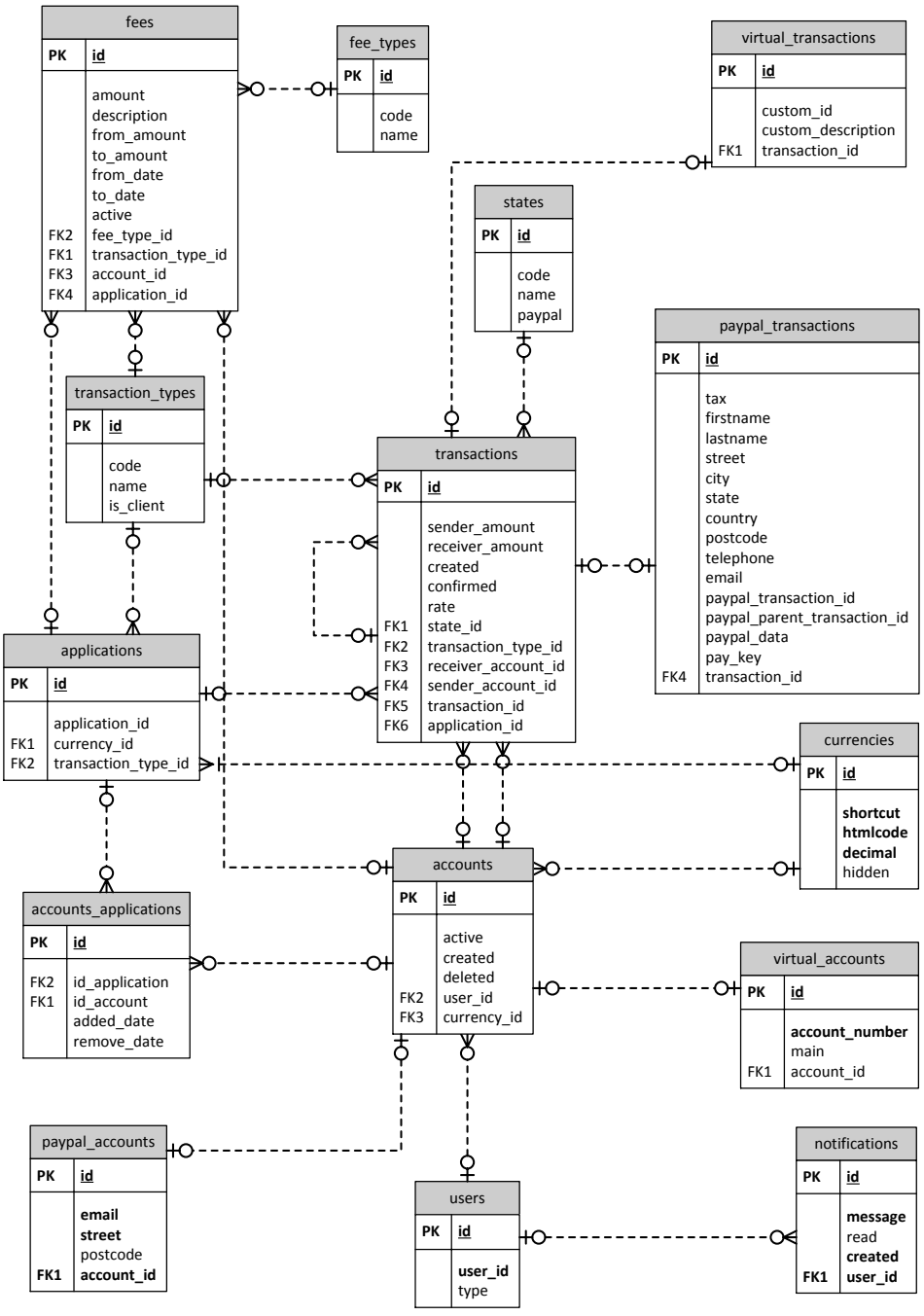
5.5.1.3 Třídní diagram znázorněný v přiloženém obrázku B, zobrazuje statický pohled na systém. V této kapitole bude popsána struktura kreditního systému.

Již samotný framework Symfony, ve kterém byl kreditní systém implementován, má předdefinovanou strukturu pro psaní webových aplikací. Jak jsem již popsal v kapitole 2.3, nejedná se o pevně danou MVC případně MVP architekturu, která je aktuálně nejpoužívanější v oblasti webových aplikací. Jako modelovou část aplikace byl zvolen ORM framework Doctrine 2.

Součástí frameworku Symfony je tzv. "Service Container"také definován jako "Dependency Injection Container". Jedná se o objekt, který je zodpovědný za vytváření instancí jiných objektů nebo-li služeb. Je však potřeba Service Container naučit, jakým způsobem dané objekty bude vytvářet včetně veškerých závislostí a jak tyto závislosti bude předávat. O vše ostatní se pak už postará sám Service Container. [28]

Struktura kreditního systému byla rozdělena do tří vrstev, které je možné vypozorovat z třídního diagramu.

Řadiče jsou zodpovědné za zachycení HTTP požadavku, který zpracuje a vrátí HTTP odpověď. Odpověď může nabývat několika typů. Může se jednat o HTML nebo XML dokument, JSON nebo například přesměrování. V případě kreditního systému se



Obrázek 4: E-R Diagram kreditního systému

jedná o první vrstvu struktury aplikace. V diagramu tříd jsou rozeznatelné podle postfixu "Controller". Jedná se tedy například o WSPaymentController, PublicController, NotificationController, VirtualAccountsController, PaypalTransactionsController a další.

Služby jsou definované již zmíněným Service Containerem, který je v řadičích plně podporován a může tak tyto služby jednoduše využívat. Většina služeb se vztahuje k dané entitě, se kterou manipuluje. Jednotlivé služby jsou proto pojmenovány podle těchto entit v jednotném čísle. Jedná se o druhou vrstvu kreditního systému.

Databázová vrstva obsahuje jednotlivé namapované entity a také "repository"třídy, ve kterých jsou předdefinovány SQL dotazy pro práci s databází ve formě metod. Každá repository se vztahuje právě k jedné entitě. K metodám repository třídy pak přistupuje právě ta služba, která je zodpovědná za manipulaci s danou entitou. Jedná se o poslední, třetí vrstvu kreditního systému.

5.6 Implementace

Jedna z hlavních fází vývoje softwaru, ve které se pomocí dokumentů, vytvořených v předešlých fázích (specifikace požadavků, analýzy a návrhu), začal kreditní systém implementovat. V této kapitole bude popsáno, jakým způsobem byly jednotlivé případy užití implementovány.

5.6.1 PayPal Vklad

Proces vkladu prostředků byl již popsán ve scénáři v kapitole (5.4.5). V této kapitole popíši, jak byl tento případ užití implementován.

Ke vkladu prostředků bylo využito BuyNow PayPal tlačítko metody PayPal Payments Standard. Jak jsem již zmínil v kapitole 3.1.1.2, tlačítko je možné vytvořit několika způsoby. Při implementaci bylo nejdříve využito vytvoření tlačíka na straně aplikace, tzn. vytvoření HTML tlačítka pomocí skrytých textových polí. Tato varianta však nebyla ideální především z důvodu bezpečnosti. Mezi skrytá pole, které jsou níže popsány, totiž patří údaje jejichž pozměnění by mohlo způsobit přinejmenším nesrovnalosti v datech.

Úmyslně uvádím, že by došlo především k nesrovnalostem v datech v tom slova smyslu, že by se provedené transakce neprojevovaly na straně kreditního systému. Nemůže tedy dojít ke zneužití této opearace a neoprávněnému získání prostředků jiného uživatele.

business Obsahuje hodnotu ve formě e-mailové adresy, podle které se kromě vlastníka daného tlačítka také identifikuje příjemce platby.

cmd Určuje typ tlačítka. Pro naše účely bylo zvoleno tlačítko BuyNow, které zastupuje hodnota "_xclick".

item_name Specifikuje, za co je platba provedena. V případě kreditního systému se vždy jednalo o vklad.

custom Hodnota tohoto pole slouží pro potřeby aplikace, do kterého se vkládá zakódované ID přihlášeného uživatele. A to z důvodu přidělení vložených prostředků, případně vytvoření paypal účtu správnému uživateli kreditního systému.

notify_url Pole obsahující URL adresu, na které se nachází můj vlastní IPN posluchač (viz. 5.6.1.1).

return Pole jehož hodnota je taktéž URL adresa. V tomto případě se však jedná o adresu, na kterou bude uživatel přesměrován po provedení platby na straně systému PayPal.

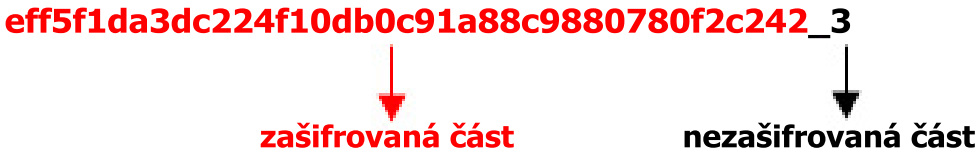
amount Pole, určující výši vkladu, kterou si volí sám uživatel.

currency_code Měna, ve které bude platba provedena.

Jak je již známo, veškeré formulářové prvky je možno libovolně změnit podstrčením vlastní hodnoty, at'se jedná o prvek skrytý či nikoliv. Mezi skrytá pole sice patří údaje jako je PayPal účet příjemce, případně plátce. Avšak pozměnění těchto polí a jejich odeslání by buď nedávalo smysl nebo by nemělo žádný účinek. V případě pozměnění účtu příjemce by došlo v nejhorším případě k tomu, že by byl "útočníkův"vklad přičten na cizí PayPal účet, nikoliv na účet superadmina což by znamenalo, že by se tento vklad v kreditním systému neprojevil. V případě pozměnění účtu plátce by nedošlo k žádnému problému, a to především z důvodu nutného přihlášení plátce na straně systému PayPal.

Dále tlačítko využíva skrytých polí jako je název položky (item_name), měna (currency_code), v jaké bude transakce provedena nebo také URL adresa notifikačních zpráv (notify_url). Změnou této adresy by však způsobilo situaci, kdy by byly prostředky zaslány na PayPal účet superadmina, ale transakce by se neprojevila na straně systému, což by způsobovalo nesrovanlosti v datech.

Posledním skrytým polem, které bylo k uskutečnění platby využito, je volitelné pole "custom", sloužící k identifikaci uživatele, který danou platbu provedl. Hodnota tohoto pole je tedy ID uživatele. V případě pozměnění tohoto pole by mohlo dojít k situaci, kdy by se vložená částka připsala na cizí primární virtuální účet. Aby se tomuto potencionálnímu problému předešlo, hodnota ID uživatele je zakódována do hash řetězce. Ten může mít tvar podobný hash řetězci na obrázku 5.6.7.



Obrázek 5: Úkazka kódování ID uživatele

Ověření tohoto ID pak probíhá při obdržení IPN zprávy obsahující tento řetězec. Nezakódovaná část se zákoduje stejným algoritmem jako ta zakódovaná a navzájem se porovnají. Pokud si tyto hash řetězce nejsou rovny, došlo k podvržení ID uživatele.

Z důvodu možného vzniku těchto nesrovnalostí byl hledán způsob, jak jim předejít. Nejdříve byla brána v úvahu varianta, kdy se po odeslání formuláře ověří vstupní hodnoty pomocí javascriptu a ajaxu, které jsou při sestavení formuláře vygenerovány. Pokud by tyto hodnoty neodpovídaly, znamenalo by to, že byly uživatelem změněny, čímž by byla operace ukončena. Tato varianta však měla nevýhodu v závislosti na javascript, který je možné v prohlížeči jednoduše vypnout. K ověření by tak vůbec nedošlo a uživateli by tak bylo umožněno pokračování v této operaci. Tomuto by bylo možné předejít například zakázáním provedení této operace v případě vypnutého javascriptu, což by však mohlo být omezující.

Systém PayPal umožňuje vytvoření tzv. šifrovaného tlačítka, které je možné pomocí PayPal API a potřebných parametrů nechat vygenerovat. Jedná se o volání API metody BMCreateButton. V HTML kódu nejsou zobrazena žádná skrytá pole, pouze klíč vygenerovaný systémem PayPal. Není tedy možné údaje podvrhnout. Jednalo se o poslední verzi tohoto tlačítka s nejvyšší mírou bezpečnosti.

Po potvrzení tlačítka je uživatel přesměrován na platební bránu PayPal. Zde jsou k dispozici dvě možnosti.

1. V případě, že má uživatel již založen vlastní PayPal účet, je požádán pouze o zadání e-mailu a hesla ke svému PayPal účtu.

2. V případě, že uživatel PayPal účet nevlastní, je vyzvám k zadání svých osobních údajů spolu s informacemi o platební kartě, mezi které patří číslo karty, datum expirace a bezpečnostní CSC kód.

5.6.1.1 IPN notifikace jsem z teoretického hlediska popsal výše v kapitole 3.1.1. V této části popíši, jakým způsobem jsem IPN notifikace implementoval a využíval v kreditním systému.

Jak už jsem zmínil výše, kreditní systém obsahuje kompletní historii jak virtuálních transakcí, tak transakcí provedených oproti platební bráně PayPal. Bylo tedy potřeba veškeré vklady a výběry peněžních prostředků evidovat v databázi kreditního systému, k čemuž slouží právě tyto IPN notifikace. Aby bylo možné odebírat IPN notifikace, bylo potřeba vytvořit již zmíněný posluchač, který na dané URL adrese jen čeká na požadavky (IPN notifikace) ze strany PayPal. URL adresu posluchače, je potřeba odeslat spolu s HTML tlačítkem pod názvem "notify_url".

Posluchač provádí dvě hlavní operace. Nejdříve provede validaci notifikace, aby se ujistil, že byla skutečně odeslána ze strany systému PayPal (viz. obrázek 1). V případě, že ověření proběhne v pořádku (PayPal vrátí odpověď VERIFIED), posluchač zpracuje příchozí data obsahující informace o provedené transakci a uloží do databáze, konkrétně do tabulky paypal_transactions s vazbou na transactions.

V případě změny stavu transakce, systém PayPal o tomto opět informuje IPN zprávou. Aby nedošlo k vícenásobnému uložení jedné transakce, kreditní systém ověří, zda-li

taková transakce již existuje. Toto ověření je provedeno pomocí ID transakce, které je součástí IPN zprávy. V případě, že taková transakce ještě evidována není, vytvoří se nová. V opačném případě se aktualizuje pouze stav a pokud se jedná o stav "Completed", nebo-li, že je transakce potvrzena, aktualizuje také datum potvrzení transakce.

5.6.1.2 Princip persistence vkladu je založen především na třech databázových tabulkách. Jedná se o tabulky:

1. transactions - obecná tabulka uchovávající společná data všech typů transakcí (prozatím PayPal a Virtuální transakce)
2. paypal_transactions - specifická a rozšiřující tabulka (potomek tabulky transactions), uchovávající data pouze pro PayPal transakce
3. virtual_transactions - specifická a rozšiřující tabulka (potomek tabulky transactions), uchovávající data pouze pro virtuální transakce

Rozšíření kreditního systému o další platební bránu by tedy znamenalo rozšíření databáze o další tabulku s vazbou na obecnou tabulku Transactions.

Cílem bylo evidovat a mít tak přehled nad všemi platbami provedenými jak mezi PayPal účty, tak mezi Virtuálními účty. Princip persistence transakcí u transakčního typu "PayPal vklad"se skládá z několika entit, které jsou na sebe navázány.

Persistence transakcí je vyvolána přijetím IPN zprávy. Tato zpráva obsahuje několik údajů potřebné pro požadovanou persistenci. Jedná se o údaje PayPal účet příjemce (superadminem vytvořený PayPal účet pro uchování peněžních prostředků získaných z vkladu případně z přijatých poplatků), PayPal účet plátce, vkladová částka, poplatek systému PayPal, ID uživatele, datum platby, status platby a ID PayPal transakce.

Entita č.1 V první entitě se eviduje částka získaná z IPN zprávy bez poplatku jak na účtu příjemce (receiver_account_id), tak na účtu plátce (sender_account_id)

Entita č.2 Druhá entita pak eviduje PayPal poplatek získaný z IPN zprávy, který je připsán na účet plátce s vazbou na předešlou paypal transakci (Entita č.1).

Entita č.3 Tato entita je evidována až v případě, že status transakce zjištěný z IPN zprávy označuje transakci za dokončenou (completed). Při splnění této podmínky se vytváří třetí entita definující pohyb mezi virtuálními účty. Při vkladu, v případě virtuální transakce, tedy plátce uveden není (tato hodnota je nastavena na NULL). Příjemci se na virtuální účet připíše částka obsažena v IPN zprávě. Může však nastat situace, kdy virtuální účet příjemce je v jiné měně, než transakce provedená systémem PayPal. Řešení je takové, že se částka u této entity přepočítá podle aktuálního kurzu ČNB. Tato transakce má taktéž vazbu na předešlou PayPal transakci (Entita č.1).

Entita č.4-n Každý typ transakce může mít přiděleno libovolné množství poplatků. Vytvoří se tedy tolik entit kolik poplatků může být uplatněno (podle definovaných podmínek).

Jak jsem se již zmínil u Entity č.3, může dojít k tomu, že příchozí transakce nemusí být bezprostředně označena za dokončenou. PayPal nabízí takovou možnost, že si příjemce (v tomto případě superadmin) může určit, zda-li požaduje veškeré platby schvalovat automaticky a nebo že je schválí sám. V druhém zmíněném případě dojde tedy k tomu, že se transakce dostane do stavu "Čeká na schválení". PayPal zašle IPN zprávu definující již zmíněný stav platby. Platební sytém tedy vytvoří pouze první 2 entity definující paypal platbu včetně poplatku. Jakmile příjemce tuto platbu potvrdí, PayPal zašle další IPN zprávu definující stav platby za již dokončenou a uloží se taktéž entity č.3 a č.4.

Příjemce tuto platbu však může i zamítnout. V tomto případě namísto stavu transakce "dokončeno" (completed), je ze strany PayPal zaslán stav "Vráceno" (reversed). Kreditní systém si s touto operací poradí tím způsobem, že původní PayPal transakci označí za dokončenou a vytvoří další 2 téměř totožné entity, u kterých se pouze zamění účet a částka plátce s účtem a částkou příjemce, dále se nastaví typ transakce (Vrácení) a nakonec i status, který je v případě vrácení označen jako "Vráceno".

Persistence těchto entit je řešena transakčně. Buď se tedy uloží všechny entity, které se uložit mají a nebo se neuloží entita žádná. Nemůže tak tedy dojít k problémové situaci, kdy by se uložila část transakce například v případě výpadku serveru apod. Pokud k takovému problému dojde, kreditní systém vrátí chybovou HTTP odpověď (HTTP kód 500) a systém PayPal tak zašle IPN zprávu znova.

Poznámka 5.2 Pro vklad kreditů je definováno jedno omezení. Jedná se o znemožnění vrácení vkladu po akceptaci platby v uživatelském rozhraní PayPal. Toto rozhraní umožňuje vrátit vklad jak před akceptací (vrácení v plné výši), tak po akceptaci, kde PayPal umožňuje i částečné vrácení. Nicméně akceptací platby se na straně kreditního systému převede vklad na uživatelův virtuální účet, což znamená, že s prostředky smí jakkoliv manipulovat. Může tedy dojít k situaci, kdy uživatel prostředky utratí a superadmin následně provede vrácení, přičemž uživatel již nedisponuje dostatečným zůstatkem. Protože je tato akce prováděna v rozhraní PayPal, který následně jen zašle informativní IPN notifikaci o provedení, kreditní systém nemá možnost dopředného provedení kontroly zůstatku.

5.6.2 PayPal Výběr

Proces výběru prostředků byl popsán ve scénáři v kapitole (D). V této kapitole popíši, jak byl tento případ užití implementován.

K výběru prostředků z kreditního systému byla využita platební metoda systému PayPal, AdaptivePayment, která umožňuje několik metod k uskutečnění platby a také několik způsobů schvalování plateb (viz 3.1.1.3). V kreditním systému je zprovozněna jednoduchá platba (simple) se dvěma způsoby schvalování. Jedná se o schvalování **explicitní**, což znamená, že je transakci potřeba potvrdit superadminem v centrálním systému, a schvalování **implicitní**, u kterého dojde ke schválení transakce automaticky bezprostředně po provedení.

V obou případech, jak u implicitního, tak i u explicitního schvalování plateb, dojde nejdříve k vytvoření požadavku na výběr určité sumy. K tomuto účelu je k dispozici

formulář obsahující 3 formulářové prvky. Jedná se o ”Částku”, kterou uživatel vyžaduje k výběru, dále pak ”Virtuální účet”, ze kterého budou prostředky strženy a nakonec ”Paypal účet”, na který bude vybraná částka odeslána. Při potvrzení formuláře je pomocí Paypal AdaptivePayments API metody ”Pay”(viz 2) získán platební klíč (PayKey).

Pokud je v kreditním systému aktivováno explicitní schvalování, uživateli se prozatím jen zablokuje vybíraná částka na zvoleném virtuálním účtu (Entita č.1,2). Transakce je tedy ve stavu ”Čeká na schválení”.

Schválení pak probíhá v centrálním systému superadministrátorem v sekci ”Transakce”právě pomocí již získaného platebního klíče (PayKey). Transakci, která je určena ke schválení náleží 2 akce. První akce slouží k potvrzení transakce, čímž je transakce uvedena do stavu ”Schváleno”, druhá naopak k zamítnutí transakce, čímž je transakce uvedena do stavu ”Zamítnuto”.

Pokud je však v kreditním systému aktivováno implicitní schvalování, transakce je schválena ihned.

V případě schválení dané transakce, ať už se jedná o explicitní či implicitní, kreditní systém obdrží IPN zprávu nesoucí informace o provedené platbě. Při zpracování této zprávy jsou vytvořeny další entity v závislosti na tom, kolik poplatků bylo k tomuto typu transakce uplatněno. Stavy všech souvisejících transakcí jsou pak změněny na stav ”Schváleno”.

Stejně jako u ostatních typů transakcí, i výběr může být složen z několika podtransakcí. Jedná se o kombinaci virtuálních a PayPal transakcí, ke kterým mohou být připojeny i poplatky (viz výčet níže). Proto bylo potřeba operaci výběru evidovat transakčním způsobem. Nesmí dojít k situaci, kdy se zaeviduje jen určitá část transakce.

Entita č.1 První entita se týká virtuální platby, která se uchová ve stavu čekání. Tímto se uživateli zablokují kredity, se kterými nemůže nadále manipulovat.

Entita č.2 Druhá entita pak definuje PayPal transakci u které se v atributu ”pay_key”(viz E-R diagram 4) uchová získaný klíč k platbě.

Entita č.3 Tato entita definuje poplatek PayPal platby a je uložena až v případě, kdy superadmin schválí transakci.

Entita č.4-n Každý typ transakce může mít přiděleno libovolné množství poplatků. Vytvoří se tedy tolik entit kolik poplatků může být uplatněno (podle definice podmínek). Stejně jako entita č.3 jsou i tyto entity evidovány až v případě, kdy superadmin potvrdí transakci

Poznámka 5.3 PayKey, nebo-li platební klíč je na straně systému PayPal časově omezen. V okamžiku, kdy tento klíč expiruje, není možné již platbu potvrdit. Životnost klíče jde však podle potřeby změnit. Při volání metody Pay je možné definovat jeden z volitelných parametrů požadavku. Jedná se o parametr ”payKeyDuration”, kterým je možné životnost nastavit v rozmezí od 5 minut do 30 dní. Formát tohoto parametru je definován jako Duration Data Type [29]. Aktuálně je v kreditním systému nastavena životnost klíče na 7 dní.

Poznámka 5.4 Stejně jako vklad, i výběr disponuje omezeními, o kterých je potřeba uživatele informovat. Může dojít k situaci, kdy po schválení výběru superadminem, je obdržena IPN zpráva nesoucí stav transakce "Pending"nebo-li stav, kdy se čeká na schválení příjemcem. Důvodem tohoto stavu může být buď "payment review"nebo-li nutnost schválení/zamítnutí transakce příjemcem a nebo "multicurrency", nebo-li zaslání na účet s jinou měnou. V těchto případech dochází k tomu, že kreditní systém není informován o tom, zda-li příjemce výběru transakci potvrdil či zamítl. Důvodem je nezaslání další IPN zprávy ze strany PayPal. V tuto chvíli tedy kreditní systém "násilně"označí transakci za schválenou. Nepočítá tedy s tím, že by uživatel požadoval vrátit vybrané prostředky kreditnímu systému.

5.6.3 Převod kreditů

jedná se o případ užití, jehož scénář je uveden v podkapitole D. Převod kreditů je jednou z dalších transakčních operací, kterou smí zastupovat jeden ze dvou možných transakčních typů uvedených v číselníku "transaction_types". Jedná se o typ "transfer"a "currency_transfer". Typ transakce se volí zcela automaticky. Vybírá se na základě zjištěných měn virtuálního účtu plátce a virtuálního účtu příjemce. Pokud jsou měny těchto účtů odlišné, kreditní systém označí transakci jako typ "currency_transfer"nebo-li převod mezi měnami. V opačném případě se pak jedná o typ "transfer". Transakce byly odlišeny z důvodu možnosti definice odlišných poplatků.

Převod je možné uskutečnit v sekci "Převody", kde je připravený formulář pro zadání potřebných údajů. Všechny tyto údaje jsou povinné a jedná se o:

Způsob převodu Aktér má možnost zvolit jeden ze dvou způsobů převodu. V závislosti na tom, zda-li aktér zná číslo účtu nebo e-mail příjemce, má možnost zvolit převod na účet nebo převod uživateli. Tato volba probíhá pomocí formulářového prvku přepínače. Je tedy možno zvolit jen jednu z těchto dvou hodnot.

Číslo účtu plátce Tento údaj ve formuláři zastupuje výběrové pole (select box) obsahující všechny aktivní účty aktéra. Z vybraného účtu tedy bude převedená výše odečtena

Číslo účtu / E-mail příjemce U tohoto údaje je pak využito textové pole, do kterého aktér vyplní buď číslo účtu nebo e-mail příjemce a to v závislosti na tom, jaký způsob převodu aktér zvolí. V případě volby převodu na uživatele, jsou kredity zaslány na primární účet příjemce podle zadaného e-mailu.

Částka Nenulová a nezáporná číselná hodnota.

Po potvrzení tohoto formuláře, kreditní systém provede několik ošetření. Stejně jako u ostatních operací manipulujícími s virtuálními účty se nejdříve zkontroluje, zda-li účet plátce je skutečně účet přihlášeného uživatele. Kreditní systém zjistí, jaký způsob převodu si uživatel zvolil a v závislosti na tom buď ověří existenci virtuálního účtu příjemce a zda-li je aktivní nebo pomocí e-mailu zjistí, o jakého uživatele se jedná a jako §čet příjemce zvolí primární účet tohoto uživatele.

Protože platební komponenta neuchovává žádné informace o uživatelích, získání entity uživatele podle e-mailu je tedy provedeno pomocí komponenty uživatelské. K této komunikaci, mezi komponentami, pak slouží rozhraní na obou stranách (viz diagram komponent 2).

Kreditní systém dále ověří účet plátce, tedy zda-li existuje a zda-li je aktivní. Také je zkontrolovám zůstatek na virtuálním účtu plátce, který musí být vyšší než částka včetně poplatků.

Pokud jsou všechny výše zmíněné podmínky splněny, jsou vytvořeny následující entity.

Entita č.1 Jedná se o hlavní entitu virtuální transakce definující převod.

Entita č.2-n Počet následujících transakcí je ovlivněn počtem uplatněných poplatků.

5.6.4 Webová služba

Jak už jsem zmínil, webová služba, poskytovaná kreditním systémem, je přístupná pomocí SOAP protokolu. Součástí webové služby je taktéž WSDL dokument (viz A) definující struktury jak vstupních tak i výstupních hodnot všech poskytovaných metod. Jednotlivé metody jsou také popsány v kapitole 5.7.1.

Kompletní systém bude využívat šifrovaného protokolu SSL (Secure Socket Layer), který však pro webové služby není dostačující. SSL zabezpečuje data na transportní vrstvě. Obsah samotné zprávy je tak ale stále případným útočníkům otevřen.

Abychom dosáhli co nejvyššího stupně zabezpeční bude v dalších fázích vývoje systému využito šifrování na úrovni zpráv zajišťující utajení a také digitálních podpisů, který posílí autenticitu, integritu a již zmíněnou nepopiratelnost.

5.6.5 Platba aplikaci

je případ užití, jehož scénář je uveden v podkapitole D. Platba aplikaci je dalším typem platebních transakcí označena kódem "application_payment". Od výše zmíněných typů se však liší především tím, že se vyvolává z klientské aplikace pomocí webové služby.

Narozdíl od výše zmíněných typů trasakcí (vklad, výběr), u tohoto typu se jedná čistě o transakci virtuální, ke které mohou být připojeny podtransakce ve formě poplatků. Poplatky jsou pak definovány adminem a superadminem v centrálním systému.

U této transakce jsou evidovány tyto entity:

Entita č.1 Jedná se o hlavní virtuální transakci definující platbu.

Entita č.2-n Počet následujících transakcí je ovlivněn počtem uplatněných poplatků.

5.6.5.1 Bezpečnost byla v případě webové služby nejdůležitejší částí. Právě z **bezpečnostních** důvodů byla platba aplikaci opatřena ověřovacím procesem. Jedná se o proces potvrzení platby uživatelem, který platbu provedl. Po provedení transakce plátce

obdrží emailovou zprávu, která kromě informací o provedené transakci obsahuje taktéž odkaz opatřený vygenerovaným hash řetězcem. Kliknutím na tento odkaz uživatel platbu potvrdí a označí ji tak za dokončenou.

Bez tohoto opatření by mohlo docházet k nechtěným platbám nebo k neoprávněnému popření provedé platby. Tímto způsobem se tyto problémové situace potlačují.

Odkaz obsažený v emailové zprávě bude mít určenou dobu životnosti. Po dosažení této životnosti bude odkaz znektivněm a transakce bude automaticky označena za zamítnutou. Tento proces bude zajišťovat tzv. cron (5.5).

Poznámka 5.5 Cron je program běžící na serveru, který automatizovaně spouští skripty v předem definovaných časech. Jedná se především o skripty s potřebou periodického spouštění. [30]

5.6.6 Vytvoření nového virtuálního účtu

Jedná se o popis implementace scénáře D. K vytvoření nového virtuálního účtu jsou nejdůležitější 2 formulářové prvky. Jedná se o číslo účtu a měnu, ve které se bude účet provozovat. Číslo účtu je definováno jako 9-místný číselný kód, který lze nechat systémem vygenerovat nebo má uživatel možnost zvolit číslo vlastní.

Formulář umožňující provedení této operace tedy obsahuje 1 přepínač (radio button), který určuje, zda-li se číslo virtuálního účtu vygeneruje nebo si jej uživatel určí sám. Ve druhém zmíněném případě se uživateli povolí zápis do textového pole označeném popiskem "Číslo účtu". Ve výchozím stavu je toto textové pole pro zápis zákázáno. Dalším prvkem, který formulář obsahuje je výběrové pole (select box), ve kterém si uživatel může vybrat jednu z několika uvedených měn. V této měně se provádí veškeré platební operace týkající se tohoto účtu s tím, že tuto měnu není možné později změnit stejně jako číslo účtu. Posledním prvkem formuláře je další výběrové pole definující aplikace, ke kterým bude tento virtuální účet přiřazen. Je možné zvolit více aplikací, jedná se tedy o multi select box. Z důvodu přívětivější uživatelské manipulace, byl zvolen jQuery plugin. Přiřazení aplikací není povinné. Je tedy možné přiřazení později změnit.

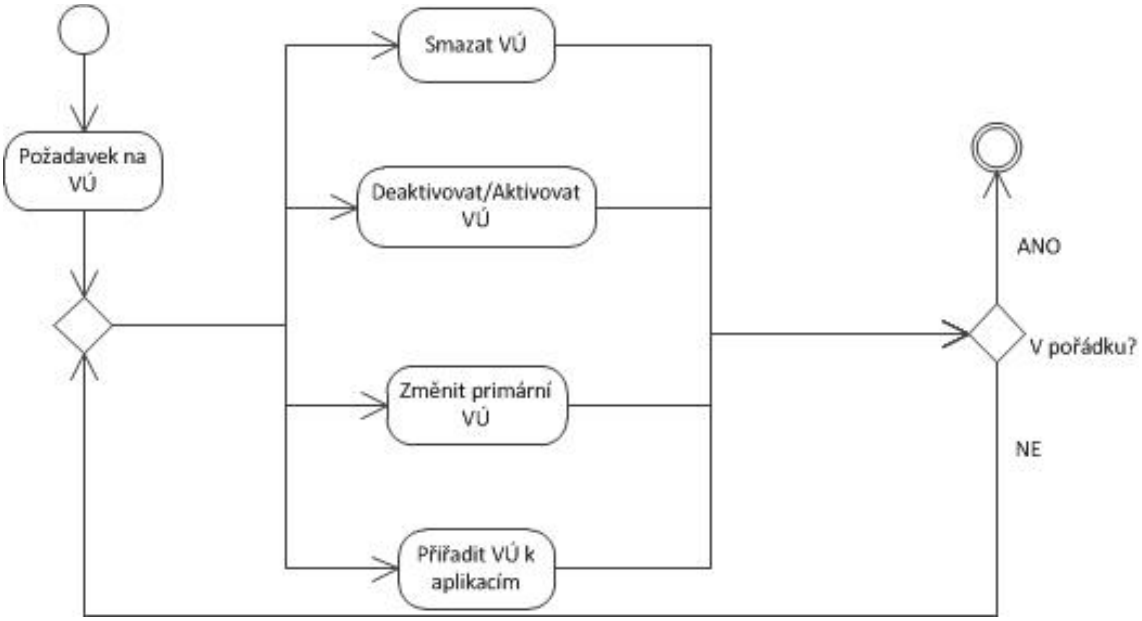
Po potvrzení formuláře kreditní systém ověří, zda-li jsou data validní. V tomto případě se tedy bude jednat o správný formát čísla účtu a ověření, zda-li nedošlo k podvržení měny za měnu systémem nepodporující. Pokud uživatel k účtu přiřadil požadované aplikace, kreditní systém provádí další kontroly popsané v podkapitole 5.6.7.1

5.6.7 Akce virtuálních účtů

Všechny operace, vstahující se k již vytvořenému virtuálnímu účtu (zrušení, aktivace, deaktivace, změna primárního účtu a konfigurace), mají z implementačního hlediska jednu společnou vlastnost. Touto vlastností je myšlen způsob, jakým se tyto akce vyvolávají.

K vyvolání této operace slouží HTML odkaz nesoucí 1 parametr. Tímto parametrem je řetězec hash definující zakódované ID daného virtuálního účtu. Princip vytváření tohoto hash řetězce je totožný s principem znázorněným na obrázku . Jedná se opět o bezpečnostní opatření, díky kterému se případnému útočníkovi znemožní provedení

operace na virtuální účet jiného vlastníka. Při vyvolání níže uvedených operací dochází ke kontrole jeho vlastníka. Proto jde především o preventivní opatření.



Obrázek 6: Přehled aktivit nad virtuálními účty

5.6.7.1 Přiřazení/odebírání virtuálního účtu aplikacím je případ užití, jehož scénář je uveden v kapitole D. Operace přiřazení virtuálního účtu k aplikacím je součástí konfigurace virtuálního účtu. Uživatel má tak možnost libovolně přiřazovat své účty k aplikacím bez jakéhokoliv omezení. Kreditní systém je navrhnut tak, aby se veškeré změny přiřazení evidovaly. Kromě toho, že je možné zpetně dohledat, kdy a ke kterému účtu měl uživatel přiřazenou danou aplikaci, tak je také možné zjistit, jaké transakce byly v konkrétním přiřazení provedeny. Veškeré tyto změny přiřazení jsou evidovány ve vazební tabulce "accounts_applications", která tvoří vazbu mezi aplikacemi (tabulka applications) a účty (tabulka accounts). (viz. E-R diagram 4). V případě odebrání účtu aplikaci se tato vazba fyzicky nesmaže. Provede se pouze aktualizace hodnoty atributu "remove_date", které se nastaví na aktuální datum. Díky tomuto atributu je možné zpětně získat historii přiřazení.

Formulář obsluhující operaci přiřazení aplikací k virtuálnímu účtu obsahuje pouze jeden prvek, a to multiselect box. Stejně jako u formuláře pro vytváření nových virtuálních účtů byl využit jQuery plugin z důvodu jednoduššího ovládání.

Po potvrzení tohoto formuláře systém provede hned několik kontrol:

1. Ověření oprávněnosti uživatele. Tedy zda-li se jedná o uživatele, kterému daný účet skutečně patří.

2. Ověření, zda-li editovaný virtuální účet je aktivní. Není možné přiřazovat deaktivovaný účet.

3. Ověření, zda-li u odebírané aplikace nemá účet blokované kredity. Je potřeba dokončit všechny platby, aby se mohl účet aplikaci odebrat.

4. Ověření, zda-li u přiřazované aplikace není již nějaký účet přiřazen. Pokud již účet přiřazen má, systém se jej pokusí odebrat, pokud je splněná podmínka z bodu 3 a následně přiřadit k požadovanému účtu.

Pokud jsou výše zmíněné podmínky splněny, účet je úspěšně přiřazen aplikacím. V opačném případě je uživatel informován o konkrétní chybě či nesplněné podmínce.

5.6.7.2 Odebrání virtuálního účtu je případ užití, jehož scénář je uveden v kapitole D. Při operaci odebrání virtuálního účtu nedochází k úplnému smazání účtu z databáze. Z historického hlediska se odebraným účtům aktualizuje pouze hodnota atributu "deleted", která se nastaví na aktuální datum.

Ke správnému provedení operace zrušení virtuálního účtu bylo definováno několik případů, které akci přeruší:

1. Ověření oprávněnosti uživatele. Tedy zda-li se jedná o uživatele, kterému daný účet skutečně patří.

2. Ověření, zda-li účet, určený ke zrušení, není primární. Kreditní Systém je navrhnut tak, aby měl vždy uživatel právě jeden primární účet. Takový účet tedy není možné odebrat. Ikona pro smazání primárního účtu není preventivně zobrazena.

3. Ověření dostupnosti jakékoliv sumy kreditů na daném účtu. Nebylo by vhodné, kdyby bylo umožněno smazat účet, na kterém jsou uloženy kredity. V případě, že systém zjistí tuto skutečnost, je uživateli nabídnuta možnost převodu kreditů na jeho primární virtuální účet a následně požadovaný účet smazat. Jedná se o usnadnění práce uživateli v případě, že převedení kreditů na primární účet je vyhovující.

V případě, že proces zrušení účtu proběhl v pořádku je kromě aktualizace atributu "deleted"odebrán také všem aplikacím. Proces odebrání účtu aplikacím je popsán v kapitole 5.6.7.1.

5.6.7.3 Aktivace/Deaktivace virtuálního účtu je případ užití, jehož scénář je uveden v kapitole D případně D. Implementaci operací aktivace a deaktivace virtuálního účtu, byla narozdíl od scénářů těchto dvou případů užití, popsána dohromady. Liší se pouze v jedné věci, a to že v případě deaktivace účtu dochází navíc také k odebrání účtu aplikacím. Tato operace je podrobněji popsána v kapitole 5.6.7.1. Jediným ověřením, které systém obstará u této operace je ověření, zda-li se jedná o vlastníka aktivovaného/deaktivovaného účtu.

Výsledkem této operace je nastavení atributu "active"v tabulce "accounts". Tento atribut je typu boolean. Očekávanými hodnotami jsou tedy 1 (virtuální účet je aktivován) nebo 0 (virtuální účet je deaktivován). Deaktivovaný účet je od aktivovaného účtu rozeznatelný jeho světle šedým podbarvením a odlišnou ikonou.

5.6.7.4 Změna primárního virtuálního účtu je případ užití, jehož scénář je uveden v kapitole D. Jedná se o poslední operaci vztahující se ke správě virtuálních účtů. Stejně jako u předchozí operace, systém provádí kontrolu, zda-li s účtem manipuluje jeho vlastník. Žádné další ošetření nebylo potřeba provádět.

Při pohledu na databázovou část je jako primární účet zvolen ten, který má nastaven atribut "main"na hodnotu 1.

Při provádění procesu změny primárního účtu je nejdříve nastaven původnímu primárnímu účtu atribut "main"na 0 a následně opačná hodnota nově určenému primárnímu účtu. Tento proces je zpracován transakčně. Nemůže tedy dojít k tomu, aby se odebral příznak původnímu primárnímu účtu a nově určenému primárnímu účtu se již nepřidělil. Vzhledově je primární účet od ostatních účtů odlišen především tučným písmem a odlišnou ikonou.

5.6.8 Sekvenční diagramy

Z důvodu rozsáhlosti bude dále v této kapitole popsán a znázorněn pouze diagram SD1 znázorňující operaci vkladu prostředků. Ostatní diagramy, SD2 - SD21, jsou včetně svých popisů uvedeny mezi přílohami viz příloha C. Mezi těmito diagramy jsou i diagramy, na které se diagram SD1 odkazuje.

K jednoduššímu a přehlednějšímu odkazování na jednotlivé bloky sekvenčních diagramů, byly tyto bloky očíslovány ((1) ... (n)). Každý diagram má pak vždy své vlastní číslování začínající číslem 1.

Jeden případ užití může být znázorněn pomocí více sekvenčních diagramů, které mezi sebou odkazují pomocí referencí uvedených v kombinovaném fragmentu s klíčovým slovem REF. Jednotlivé diagramy byly rozděleny podle jejich významu a to buď z důvodu nedostatku prostoru a nebo z důvodu možnosti znovupoužití.

SD1: Vklad Tento diagram obsahuje podrobnější informace týkající se komunikace mezi objekty, což způsobilo větší rozsáhlost v porovnání s ostatními diagramy. Dohromady je tedy složen z 9 vzájemně propojených diagramů.

Základním diagramem operace "Vklad"je diagram č.1 (SD1), který znázorňuje komunikaci 4 objektů (jedná se o instance tříd IPNListenerController, IPNListener, PayPalAPICaller a Transaction). Výchozím objektem je instance třídy IPNListenerController, který přijímá požadavek (IPN zpráva) odeslaný systémem PayPal. Výstupem je pak HTTP odpověď s kódem 200 nebo s kódem 500 v případě zachycení vyjímky.

(1) V první fázi dochází k validaci příchozí IPN zprávy, jejimž výsledkem je buď hodnota TRUE v případě validní zprávy a nebo je vyhozena vyjímka s chybovou hláškou oznamující nevaliditu zprávy.

(2) Pomocí metody prepareDepositTransaction() dochází k přípravě PayPal transakce viz diagram 8.

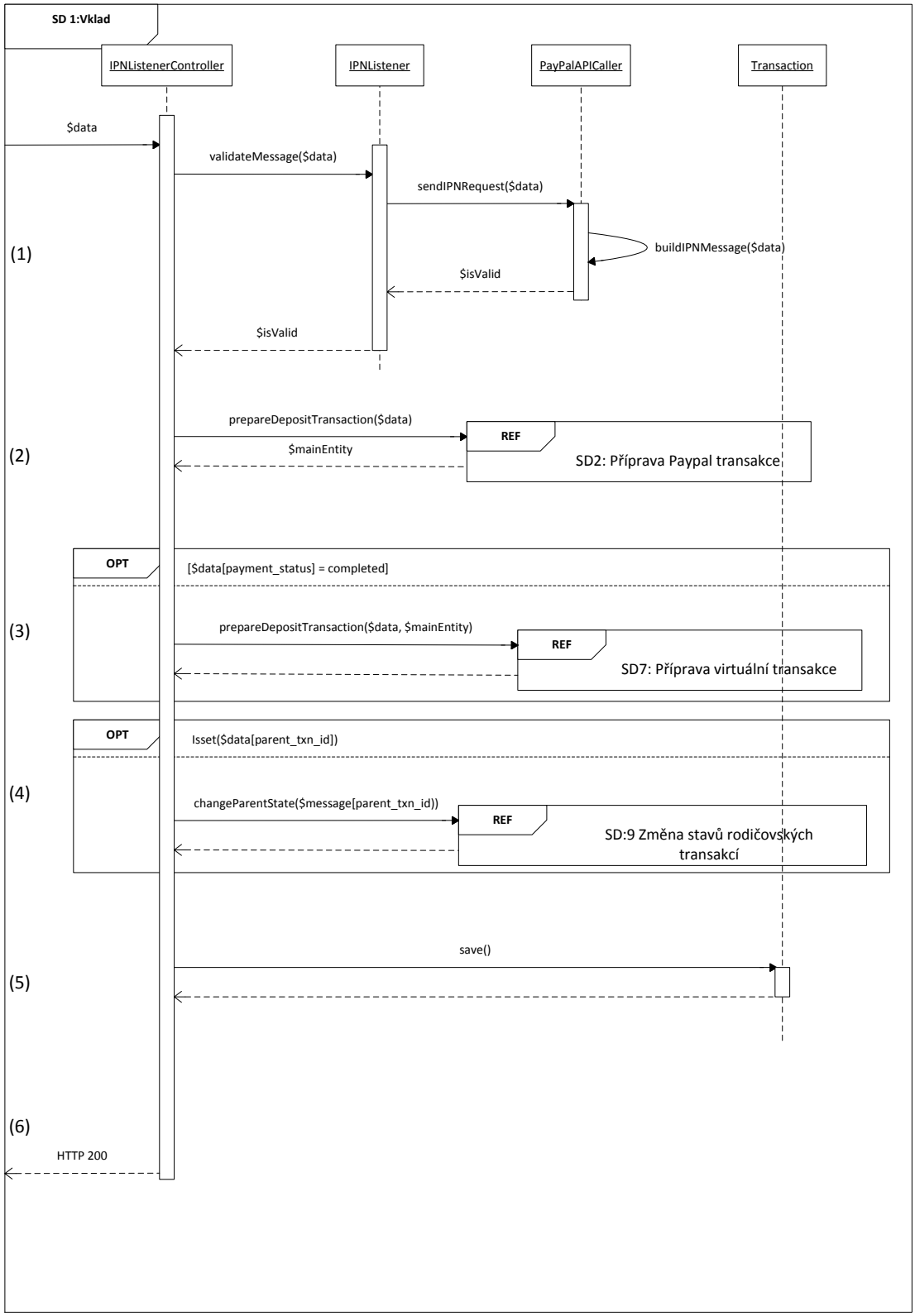
(3) Pokud je platba ve stavu "completed", provede se příprava virtuální transakce viz diagram 9.

(4) Pokud příchozí zpráva IPN obsahuje rodiče transakce (parent_txn_id), je provedena změna stavů i u rodičovských transakcí v kreditním systému viz diagram 10.

(5) Všechny připravené transakce se pomocí metody save() transakčně uloží.

(6) Jako návratová hodnota úspěšně dokončené operace je prázdná odpověď s HTTP kódem 200, který PayPalu slouží jako potvrzení, že IPN zpráva byla přijata a zpracována.

Co však v diagramu zakresleno není, je zachycení a zpracování vyjímky. V takovém případě je výstupem operace prázdná odpověď s HTTP kódem 500. PayPal tedy zasílá IPN zprávu opakovaně.



Obrázek 7: SD1: Vklad

5.7 Popis rozhraní komponenty

Jak jsem již zmínil výše, kreditní systém disponuje několika rozhraními. Tato rozhraní jsou přehledně znázorněna v diagramu komponent (viz obrázek 2). V této kapitole pak budou detailně popsány jednotlivé metody všech rozhraní.

5.7.1 Rozhraní webové služby

Výše jsem již uvedl, že jako rozhraní webové služby je určena třída SFWSPaymentFacade. Jedná se tedy o fasádu, která je závislá na jiných třídách kreditního systému. Díky těmto třídám je možné webové službě vystavit jakoukoliv funkčnost kreditního systému a to zcela odděleně.

Ke komunikaci mezi klientskou aplikací a webovou službou je využito protokolu SOAP 2.6. U každé z níže uvedených metod bude vždy pouze jeden vstupní parametr, který je však typu object a může tak obsahovat více vlastostí. Je to dáno tím, že SOAP transformuje jednolivé vlastnosti objektů na elementy XML dokumentu. Taktéž je namísto objektu možné využít asociativního pole, nicméně z mého pohledu je přehlednější využití objektu. Rozhraní webové služby je pak popsáno pomocí WSDL dokumentu 2.6.0.1 (A), který zveřejňuje a popisuje tyto metody:

1. getAmount()

   - **Popis:** Metoda getAmount slouží k získání zůstatku a také počtu blokovaných kreditů daného uživatele
   - **Vstupní parametry:** object $request nesoucí jen jednu vlastnost
     - user (povinný) - definuje konkrétního uživatele (uživatel je však přenášen ve formě tokenu, který klientské aplikaci vystaví uživatelský systém. Jedná se o konvenci mezi kreditním a uživatelským systémem) (typ: string)
   - **Návratová hodnota:** object $response obsahuje dohromady 5 vlastností:
     - balance (povinný) - zůstatek (typ: decimal)
     - blocked (povinný) - počet blokovaných kreditů (typ: decimal)
     - error (povinný) - určuje, zda-li se vyskytla chyba (typ: boolean)
     - code - určuje kód chyby (typ: integer)
     - message - zpráva popisující chybu (typ: string)

```
<soapenv:Body>
    <urn:getAmount soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        >
        <urn:getAmountRequest>
            <user xsi:type="xsd:string">YToyOntzOjU6InRva2V...</user>
        </urn:getAmountRequest>
    </urn:getAmount>
</soapenv:Body>
```

Výpis 1: Příklad požadavku metody getAmount()

```
<SOAP−ENV:Body>
  <ns1:getAmountResponse>
    <response xsi:type="SOAP−ENC:Struct">
      <balance xsi:type="xsd:int">200</balance>
      <blocked xsi:type="xsd:float">54</blocked>
      <error xsi:type="xsd:boolean">false</error>
      <code xsi:type="xsd:int">0</code>
      <message xsi:type="xsd:string"></message>
    </response>
  </ns1:getAmountResponse>
</SOAP−ENV:Body>
```

Výpis 2: Příklad odpovědi metody getAmount()

2. proceedAppPayment()

   - Popis: Pomocí této metody je možné uskutečňovat platby v klientských aplikací
   - Vstupní parametry: object $request nesoucí 3 vlastnosti
     - payer (povinný) - token, podle kterého kreditní systém rozpozná plátce (typ: string)
     - amount (povinný) - částka k zaplacení (typ: decimal)
     - custom - libovolná vlastní data, která se vrátí v odpovědi (typ: string)
   - Návratová hodnota: object $response obsahuje 4 vlastností:
     - transaction (nepovinný) - vytvořená transakce (uživatelská aplikace si ji tak může evidovat ve vlastní databázi) (typ: object)
     - error (povinný) - určuje, zda-li se vyskytla chyba (typ: boolean)
     - code - určuje kód chyby (typ: integer)
     - message - zpráva popisující chybu (typ: string)

```
<soapenv:Body>
  <urn:proceedAppPayment soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/
      encoding/">
    <parameters xsi:type="urn:proceedPaymentRequest">
      <payer xsi:type="xsd:string">YToyOntzOjU6InRva2V...</payer>
      <amount xsi:type="xsd:decimal">10</amount>
      <!−−Optional:−−>
      <custom xsi:type="xsd:string">test</custom>
    </parameters>
  </urn:proceedAppPayment>
</soapenv:Body>
```

Výpis 3: Příklad požadavku metody proceedAppPayment()

```
<SOAP−ENV:Body>
  <ns1:proceedAppPaymentResponse>
    <return xsi:type="ns1:proceedPaymentResponse">
      <error xsi:type="xsd:boolean">false</error>
```

```
                <code xsi:type="xsd:integer">0</code>
                <message xsi:type="xsd:string"></message>
                <transaction xsi:type="ns1:transaction">
                    <id xsi:type="xsd:integer">352</id>
                    <custom xsi:type="xsd:string">demand</custom>
                    <state xsi:type="xsd:string">pending</state>
                    <transaction_type xsi:type="xsd:string">
                        application_payment
                    </transaction_type>
                    <payer_account xsi:type="xsd:string">test4@test.cz</payer_account>
                    <payer_amount xsi:type="xsd:decimal">1</payer_amount>
                    <receiver_amount xsi:type="xsd:decimal">1</receiver_amount>
                </transaction>
            </return>
        </ns1:proceedAppPaymentResponse>
    </SOAP-ENV:Body>
```

Výpis 4: Příklad odpovědi metody proceedAppPayment()

3. getVirtualAccount()

   - Popis: Každý uživatel má možnost přidělit jednotlivým aplikacím jeden ze svých účtů. Tato metoda vrátí virtuální účet, který má uživatel přidělen k dané aplikaci.
   - Vstupní parametry: object $request nesoucí jen jednu vlastnost
     - user (povinný) - definuje konkrétního uživatele (uživatel je však přenášen ve formě tokenu, který klientské aplikaci vystaví uživatelský systém. Jedná se o konvenci mezi kreditním a uživatelským systémem) (typ: string)
   - Návratová hodnota: object $response obsahuje 4 vlastnosti:
     - accountNumber (povinný) - číslo virtuální účtu (typ: string)
     - error (povinný) - určuje, zda-li se vyskytla chyba (typ: boolean)
     - code (povinný) - určuje kód chyby (typ: integer)
     - message (povinný) - zpráva popisující chybu (typ: string)

```
<soapenv:Body>
  <urn:getVirtualAccount soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <urn:getVirtualAccountRequest>
      <user xsi:type="xsd:string">YToyOntzOjU6InRva2V...</user>
    </urn:getVirtualAccountRequest>
  </urn:getVirtualAccount>
</soapenv:Body>
```

Výpis 5: Příklad požadavku metody getVirtualAccount()

```
<SOAP-ENV:Body>
  <ns1:getVirtualAccountResponse>
    <response xsi:type="SOAP-ENC:Struct">
```

```
            <accountNumber xsi:type="xsd:string">654305499</accountNumber>
            <error xsi:type="xsd:boolean">false</error>
            <code xsi:type="xsd:int">0</code>
            <message xsi:type="xsd:string"></message>
        </response>
    </ns1:getVirtualAccountResponse>
</SOAP-ENV:Body>
```

Výpis 6: Příklad odpovědi metody getVirtualAccount()

Poznámka 5.6 Výstupy všech následujících metod mají společné především to, že nabývají typu asociativního pole obsahující převážně 2 indexy. První index "result"definuje, zda-li při operaci došlo k chybě, či nikoliv. Nabývá tedy hodnot TRUE/FALSE. Pokud k chybě nedošlo, 2. index je již volitelný a obsahuje požadovaný výstup. V případě vzniku chyby, je druhý index pojmenován "message"a jeho hodnout je krátká zpráva, definující vzniklý problém. V následujících popisech rozhraní se budu věnovat převážné úspěšným scénářům, tedy druhému indexu korektně provedené operace.

5.7.2 Rozhraní komunikace s centrálním systémem

Centrální systém hraje významnou roli co se týče nastavení kreditního systému. Veškerá konfigurace probíhá právě zde. Aby se funkčnost, využívající kreditním systémem, oddělila od zbytku aplikace, bylo opět připraveno rozhraní definující potřebné operace.

5.7.2.1 Adaptér mezi kreditním a centrálním systémem je třída, pojmenovaná CentralAdapter, adaptující metody centrálního systému na vlastní metody kreditního systému. Třída je tedy součástí kreditního systému. Pokud kreditní systém vyžaduje cokoliv od systému centrálního, komunikace probíhá jen a pouze přes tuto třídu.

Aktuálně třída obsahuje tyto metody:

1. getApplicationBy($filter)
    - **Popis:** Metoda která získá jednu entitu aplikace podle zadaného filtru
    - **Vstupní parametry:** *array $filter* Asociativní pole ($key =¿ $value páry), kde key je název atributu a value je jeho hodnota
    - **Návratová hodnota:** *Application $application* Požadovaná entita
2. getApplications()
    - **Popis:** Vrátí seznam aplikací (využívá se například ve formuláři pro vytvoření nového virtuálního účtu)
    - **Vstupní parametry:** Žádné
    - **Návratová hodnota:** *array() Application* Asociativní pole obsahující entity získáné z centrálního systému. Každá položka pole je objekt typu Application.

5.7.2.2 Fasáda mezi kreditním a centrálním systémem

Jedná se o třídu "CentralFacade", která slouží pouze pro potřeby centrálního systému. Pomocí této třídy bude centrální systém konfigurovat platební systém.

1. newApplication($obj)
   - **Popis:** Vytvoří záznam nové aplikace, které následně vygeneruje a přiřadí virtuální účet
   - **Vstupní parametry:** *object $obj* Object obsahující vlastnosti applicationId (ID aplikace v centrálním systému), currencyId (ID zvolené měny), type (Vybraný typ plateb) a componentUserId (ID uživatele z uživatelského systému)
   - **Návratová hodnota:** *$account* Vygenerovaný účet aplikace
2. getCurrencies()
   - **Popis:** Vrátí seznam měn, které kreditní systém zveřejňuje
   - **Vstupní parametry:** Žádné
   - **Návratová hodnota:** *array() Currencies* Pole měn typu Currencies
3. getSelectTransactionTypes($client = FALSE)
   - **Popis:** Vrátí seznam typů transakcí, které je možné přiřazovat zaregistrovaným aplikacím.
   - **Vstupní parametry:** *boolean $client* Příznak podle kterého jsou dohledány buď všechny typy transakcí a nebo jen ty, které jsou určené pro klientské aplikace.
   - **Návratová hodnota:** *array() Application* Asociativní pole obsahující typy transakcí
4. getSelectFeeTypes()
   - **Popis:** Vrátí seznam všech typů poplatků.
   - **Vstupní parametry:** Žádné
   - **Návratová hodnota:** *array()* Asociativní pole obsahující typy poplatků
5. getApplicationFees($applicationId)
   - **Popis:** Vrátí seznam všech poplatků dané aplikace.
   - **Vstupní parametry:** *integer $applicationId* Jednoznačný identifikátor aplikace
   - **Návratová hodnota:** *array()* Asociativní pole obsahující poplatky dané aplikace
6. getFees()
   - **Popis:** Vrátí seznam všech poplatků.
   - **Vstupní parametry:** Žádné

- **Návratová hodnota:** *array()* Asociativní pole obsahující všechny poplatky

7. getFee($id)
   - **Popis:** Vrátí entitu požadovaného poplatku
   - **Vstupní parametry:** *integer $id* Jednoznačný identifikátor poplatku
   - **Návratová hodnota:** *array()* Asociativní pole obsahující všechny poplatky
8. getTransactions($applicationId = NULL)
   - **Popis:** Metoda, která centrálnímu systému vrátí buď výpis všech provedených transakcí a nebo v případě uvedení nepovinného parametru $applicationId pouze transakce dané aplikace.
   - **Vstupní parametry:** *integer $applicationId* (nepovinný) Jednoznačný identifikátor aplikace
   - **Návratová hodnota:** *array()* Asociativní pole obsahující transakce
9. getApplicationVirtualAccount($applicationId)
   - **Popis:** Metoda která kromě vygenerovaného virtuálnío účtu uvedené aplikace vrátí taktéž zůstatek a blokované kredity na tomto účtu
   - **Vstupní parametry:** *integer $applicationId* Jednoznačný identifikátor aplikace
   - **Návratová hodnota:** *array()* Asociativní pole obsahující entitu virtuálního účtu, zůstatek a sumu blokovaných kreditů
10. getPrimaryVirtualAccount($userId)
    - **Popis:** Pomocí této metody je možné získat primární virtuální účet včeteně zůstatku a počtu blokovaných kreditů podle jednoznačného identifikátoru uživatele
    - **Vstupní parametry:** *integer $applicationId* Originální jednoznačný identifikátor uživatele
    - **Návratová hodnota:** *array()* Asociativní pole obsahující entitu primárního virtuálního účtu, zůstatek a sumu blokovaných kreditů
11. saveFee($data, $applicationId)
    - **Popis:** Metoda, pomocí které je možné ukládat nové poplatky nebo editovat stávající.
      - *$data* Data získaná z potvrzeného formuláře
      - *$applicationId* Jednoznačný identifikátor aplikace
      - *$userId* Jednoznačný identifikátor uživatele
      - *$feeId* Jednoznačný identifikátor poplatku
    - **Návratová hodnota:** *array()* Asociativní pole obsahující entitu primárního virtuálního účtu, zůstatek a sumu blokovaných kreditů

5.7.3 Rozhraní komunikace s uživatelským systémem

Uživatelský systém je kreditním systémem využit především jako autorizační a autentifikační služba. V případě těchto dvou systému se jedná o jednostrannou komunikaci, kde kreditní systém využívá uživatelský, nikoliv naopak. Z tohoto důvodu je využito pouze jednoho rozhraní na obou stranách. Na straně kreditního systému se jedná o adaptér, na straně uživatelského systému pak o fasádu (viz. 2).

5.7.3.1 Adaptér mezi kreditním a uživatelský systémem Jedná se o třídu UserAdapter, která slouží ke komunikace s uživatelskou komponentou. Prozatím jsou využity tyto metody:

1. getCustomer($usertoken)
   - **Popis:** Vrátí uživatele podle tokenu, který jednoznačně specifikuje daného uživatele
   - **Vstupní parametry:** *string $usertoken* Token
   - **Návratová hodnota:** *Customer $user* Entita uživatele typu Customer

5.7.4 Rozhraní ke komunikaci se systémem PayPal

Jak je možné vidět na diagramu komponent (2), kreditní systém využívá 2 typy rozhraní pro komunikaci s PayPal systémem. Tato rozhraní slouží především k evidenci jednotlivých PayPal transakcí (vklad, výběr) v kreditní systému.

IPNListenerController Listener nebo-li posluchač je nezbytnou součástí aplikace, díký které je možné evidovat jednotlivé PayPal platby v aplikaci. Komunikace a také princip byl popsán výše v kapitole 3.1.1. Vzhledem k tomu, že posluchač vyčkává na požadavek, který nadále zpracovává, bylo potřeba rozhraní umístit do controlleru. V této podkapitole popíši metody, které tuto funkčnost obstarávají. Jedná se o 2 metody:

1. buyNowButtonIPN(Request $request)
   - **Popis:** Odposlechne příchozí požadavek, obsahující detaily o provedeném vkladu, který následně zpracuje a zaeviduje
   - **Vstupní parametry:** *Request $request* Požadavek obsahující data transakce
   - **Návratová hodnota:** Vrátí prázdnou odpověď s HTTP kódem 200 nebo 500 (v případě chyby)
2. adaptivePaymentsIPN(Request $request)
   - **Popis:** Odposlechne příchozí požadavek, obsahující detaily o provedeném výběru, který následně zpracuje a zaeviduje
   - **Vstupní parametry:** *Request $request* Požadavek obsahující data transakce
   - **Návratová hodnota:** Vrátí prázdnou odpověď s HTTP kódem 200 nebo 500 (v případě chyby)

PaypalAPICaller Třída PaypalAPICaller slouží jako rozhraní odesílající požadavky na platební bránu Paypal, konkrétně Paypal Classic API, ze kterého kreditní systém využívá hned několik metod.

1. sendValidationRequest($data, $test)
   - **Popis:** Odesílá požadavek určený k validaci platby (viz. 3.1.1.2. Odeslání validačního požadaveku vyžaduje Paypal z důvodu bezpečnosti.
   - **Vstupní parametry:**
     - *$data* Originální zpráva bez jakékoliv změny
     - *$test* Definuje, zda-li se jedná o testovací nebo ostrý požadavek. Podle toho se zvolí url adresa příjemce požadavku.
   - **Návratová hodnota:** boolean $isValid Vrátí TRUE v případě validní zprávy. V opačném případě je vyhozena vyjímka.
2. sendPayApiRequest($message)
   - **Popis:** Metoda využívaná pří operaci "Výběr", pomocí které je získán klíč uživatelem vyvolané platby. Klíč nadále využívá superadmin pro potvrzení platby.
   - **Vstupní parametry:** *$message* Sestavéná zpráva s požadovanými parametry
   - **Návratová hodnota:** object $data Objekt obsahující základní informace o provedné platbě. V případě chyby je vyhozena vyjímka.
3. sendPaymentDetailApiRequest($paykey)
   - **Popis:** Pomocí této metody kreditní systém získá bližší informace o transakci s uvedeným klíčem platby (vstupní parameter). Je využita pro zjištění stavu transakce a ID transakce potřebného pro další operaci
   - **Vstupní parametry:** *$paykey* klíč platby
   - **Návratová hodnota:** object $data Objekt obsahující detailní informace o platbě. V případě chyby je vyhozena vyjímka.
4. sendTransactionDetailRequest($transactionId)
   - **Popis:** Tato metoda je volána pouze z důvodu získání výše poplatku dané transakce
   - **Vstupní parametry:** *$transactionId* ID transakce získané z výše uvedené metody
   - **Návratová hodnota:** array $output Asociativní pole obsahující informace o platbě. V případě chyby je vyhozena vyjímka.

5.8 Zabezpečení

Jak jsem již zmínil v předešlých kapitolách, kreditní systém poskytuje několik typů transakcí určitým způsobem manipulujícími s kredity uživatelů. Právě takové operace jsou velikým lákadlem pro útočníky s vidinou získání těchto peněžních prostředků. Ať už se

jedná o operace, které se klientovi nabízí jako možnost k integraci přímo do vlastního systému a nebo operace, která se provádí v samotném kreditním systému. Většina těchto operací je obsluhována klasickými HTML formuláři, vyžadující zadání vstupních dat od uživatele, které se ve výchozím(nezabezpečeném) stavu nabízí jako jednoduše napadnutelné. Tyto útoky stojí většinou na základě podvržení systémových data formuláře za data vlastní, přičemž takto odeslaný formulář by v případě neošetření mohl provést operaci, ke které není daný uživatel oprávněný. Jde o tzv. CSRF útok popsaný v kapitole 5.8.1.

Dobrým příkladem může být převod kreditů z jednoho virtuálního účtu na jiný účet. V rámci kreditního systému je systémem požadováno provedení vstupů jako výběr jednoho z vlastních virtuálních účtů, odkud budou finance odeslány, dále cílový účet, na který budou finance připsány a nakonec musí uživatel zadat čátku, kterou k převedení vyžaduje. V případě, že by formulář neobsahoval žádné zabezpečení, útočník by měl tu možnost jednoduše otevřít zdrojový kód stránky, odkud by si formulář zkopíroval a vložil si jej například do své aplikace. Ve své aplikaci by si pak zkopírovaný formulář upravil podle vlastních potřeb a následně by měl možnost formulář libovolně odesílat. V případě operace převodu kreditů mezi účtu by tedy mohlo dojít k tomu, že si jako zdrojový účet uživatel zvolí účet jiného uživatele, který může buď znát nebo jej zkusí odhadnout, a jako cílový virtuální účet pak zvolí svůj vlastní. Samozřejmě pravděpodobnost odhadnutí 9-ti ciferného čísla je velmi nízká, ale nebylo řečeno, že si útočník nemůže vytvořit skript, který mu bude v cyklu dané číslo virtuálního účtu například inkrementovat a formulář odesílat, čímž se pravděpodobnost uhodnutí virtuálního čísla účtu značně zvýší.

Výše zmíněný útok je pro útočníka snad nejjednodušeji proveditelný, samozřejmě jen v případě, že na to aplikace není připravena a neprovedla žádná bezpečnostní opatření. Nejjednodušší je především z toho důvodu, že sám útočník má přístup do dané aplikace, tedy i k dané operaci, takže nepotřebuje zneužít například přístupu jiných uživatelů. Taktéž zabezpečení tohoto útoku nevyžadovalo žádné složité metody. Na straně kreditního systému, postačilo vstup pro zadání zdrojového čísla účtu podmínit pouze na účty přihlášeného uživatele. Avšak výše zmíněné zabezpečení není dostatečné. A to v případě, že se CSRF útok provede na základě jiného uživatele.

Jiný případ CSRF útoku může nastat, kdy uživatel přístup do kreditního systému nemá. Taktéž uvedu jako příklad operaci převodu kreditů z jednoho virtuálního účtu na druhý. V tomto případě by už ale útočník nenapadal kreditní systém za cílem získání kreditů jiných uživatelů, protože nevlastní svůj virtuální účet, ze kterého by si prostředky později vybral na svůj bankovní účet, ale především proto, aby v systému provedl jakékoliv škody (smazání záznamů, editace apod.). To však útočník musí znát strukturu systému nebo-li kam se dané požadavky odesílají a s jakými hodnotami.

Proti výše zmíněným útokům byl a nebo v pozdější fázi vývoje bude kreditní systém zabezpečen několika způsoby. V první řadě bude systém zabezpečen před možností odposlouchávání komunikace mezi prohlížečem a webovým server využitím šifrovaného síťového protokolu HTTPS, kde jsou data šifrována protokolem SSL. Dalším opatřením bylo provádění veškerých operací HTTP metodou POST namísto metody GET, kde druhá zmíněná metoda je jednodušeji napadnutelná už z toho důvodu, že jsou data posílány v URL adrese. Proti CSRF útokům pak každý formulář v kreditním systému obsahuje

tzv. podpis nebo-li token, který se náhodně generuje při každém vykreslení formuláře a jehož hodnota je zakódována pomocí hashovací funkce SHA1 v závislosti na aktuálním datu včetně času, sessionID (viz poznámka 5.7) přihlášeného uživatele a staticky zvolené 32-znakové kombinaci.

V kreditním systému má uživatel také možnost manipulovat s virtuálními účty pomocí operací jako je odebrání, aktivace, deaktivace nebo například změna primárního účtu. Nesmí však být umožněna manipulace s cizími účty. Vzhledem k tomu, že se tyto operace provádí pomocí HTML odkazu, je potřeba systému předat aktuální účet, na kterém se má operace provést. V URL adrese je tedy při operaci uveden jednoznačný identifikátor (ID) daného účtu. V případě nezabezpečení by bylo možné toto ID změnit, čímž by se operace provedla na jakémkoliv jiném účtu. Aby se tomuto předešlo, bylo využito preventivně hned dvou opatření. V první řadě je ID uveden v URL adrese zašifrováno pomocí hashovací funkce SHA1. Princip funkcionality tohoto opatření jsem však již znázornil na obrázku 5.6.7 včetně níže uvedeného popisu. Druhým opatřením pak je kontrola, zda-li uvedený uživatel, provádějící tuto operaci je skutečně vlastník daného účtu.

Poznámka 5.7 SessionID neboli SID je náhodně vygenerovaná unikátní posloupnost znaků obsahující jak čísla, tak i písmena, pod kterou se uživatelé identifikují. Nejčastěji se SID ukládá do cookie prohlížeče, což je bezpečnější řešení než ukládání například do URL adresy. Avšak nevýhodou ukládání do cookie je především ta, že ne každý uživatel má cookie zapnutou. Procento takových uživatelů je však zanedbatelné a přednost se dává tedy většinou bezpečnosti.

5.8.1 CSRF útok

CSRF útok nebo-li Cross-Site Request Forgery, často nazývané jako XSRF, je jeden z několika možných způsobů napadení webových aplikací. Jedná se o neočekávané vytvoření požadavku na aplikaci za účelem provedení neautorizované operace. Pomocí tohoto útoku je většinou možné provést operaci pod jiným přihlášeným uživatelem, který je k provedení operace oprávněn.

Takovýto útok je možné provést například pomocí HTML tagu "img"ve spolupráci s atributem "src". To však jen v případě, že aplikace, kterou útočník napadá, provádí své operace metodou GET. Tedy všechny parametry jsou uvedeny v URL adrese. Po vytvoření takového obrázku pak útočníkovi zbývá přimět jakéhokoliv uživatele, přihlášeného u aplikace, jenž se chystá napadnout, aby si tento obrázek zobrazil, nebo-li na něj kliknul. Tímto způsobem uživatel odešle požadavek aplikaci, aniž by o tom věděl.

Tento způsob se dá ošetřit záměnou GET metody za metodu POST, ale ani ta není úplně spolehlivá. V kombinaci s javascriptem, se dá i tato metoda prolomit. A to například vytvořením skrytého formuláře, který se odešle ihned po načtení stránky.

Spolehlivým a dostatečným řešením bývá využití právě metody POST v kombinaci se skrytým prvkem formuláře, obsahující vygenerovaný řetězec. Tomuto skrytému prvku se často říká token, který zná pouze aplikace. Token je většinou uložen buď v session

prohlížeče nebo v databázi, na základě čehož se token v příchozím požadavku porovnává. Tento způsob je zprovozněn u všech formulářů kreditního systému.

5.8.2 Zabezpečení webové služby

Webová služba, která je součástí kreditního systému může být taktéž cílem mnoha útoků. Aby nedocházelo k provádění nechtěných platebních transakcí, uskutečněných v klientské aplikaci, je při každém požadavku k platbě zaslán e-mail kreditním systémem, který uživatele informuje o provedené transakci a taktéž jej vyzývá k jeho potvrzení. Pokud si je uživatel vědom toho, že platbu provedl skutečně on, má možnost transakci potvrdit. Pokud však potvrzení neprovede do určité doby, platba bude automaticky zamítnuta.

Každá klientská aplikace, ke které je v centrálním systému administráorem zaregistrován kreditní systém, obdrží unikátní ID aplikace a vegenerované heslo. Z bezpečnostního hlediska nejsou tyto citlivé údaje zasílány e-mailem nebo jinou cestou. Administrátor má však možnost tyto údaje vyzvednout přímo v centrálním systému u zaregistrované aplikace.

Pomocí těchto údajů se pak při každém požadavku daná aplikace autentifikuje zasláním těchto údajů v hlavičce SOAP zprávy. Kreditní systém má pak téměř úplnou jistotu, že je webová služba využívaná oprávněným klientem. Úmyslně však uvádím "téměř úplnou jistotu"a to z toho důvodu, že v případě nezabezpečené komunikace by mohlo dojít k odchycení těchto údajů a následnému zneužití služby. Webová služba využívá k výměně zpráv protokol SOAP, jenž komunikuje přes HTTP nebo v naše případě se bude jednat o HTTPS protokol.

Dalším postupem pro zabezpečení této služby bude především zašifrování zasílané zprávy bez kterého by měl případný útočník, kterému by se podařilo prolomit zabezpečenou transportní vrstvu (HTTPS), přístup k citlivým datům obsažených v této zprávě.

5.8.3 Validace formulářů

Validace formuláře probíhá jak na straně serveru, tak i na straně klienta pomocí jQuery knihovny validate. Tato knihovna dokáže uživatele upozornit na neplatný vstup bezprostředně po jeho zadání. Není tak potřeba formulář potvrzovat a čekat na přenačtení stránky. Validace na straně serveru je především z bezpečnostních důvodů a to především pro případ vypnutého javascriptu v prohlížeči, čímž je použitá jQuery validace, určená především k přívětivější manipulaci, na straně klienta nefunkční.

6 Centrální systém

O centrálním systému bylo v předešlých kapitolách zmíněno již několikrát a vzhledem k tomu, že se jedná o centrální uzel všech komponent, bude tento systém alespoň obecně popsán v této kapitole.

V diagramu komponent (viz obrázek 2) je znázorněno napojení prozatím dvou komponent na centrální systém. Jedná se o kreditní systém a systém uživatelský. Centrální systém je stejně jako tyto 2 zmíněné systémy ve stavu vývoje a postupně se rozšiřuje o další funkcionalitu.

Záměrem tohoto systému je vytvoření prezentační neboli veřejné části webu, která bude sloužit mimo jiné k prezentaci komponent a k registraci webových uživatelů případně administrátorů. Druhá část pak bude neveřejná, do které budou mít přístup jen zaregistrovaní administrátoři a superadministrátor. Zaregistrovaní weboví uživatelé budou mít přístup pouze do uživatelských rozhraní jednotlivých komponent, nikoliv do centrálního systému.

Roli superadmin je v centrálním systému aktuálně umožněna správa webových uživatelů a administrátorů, dále pak přehled nad všemi aplikacemi a správa komponent k nim zaregistrovaným. Co se týče kreditního systém, superadmin má k dispozic přehled transakcí všech aplikací, dále pak přehled nad svým virtuálním účtem, který prozatím slouží k příjmu poplatků z plateb. Superadmin má také k dispozici úplnou správu poplatků a to jak globálních, týkající se všech aplikací, tak specifických pro danou aplikaci.

Role admin má pak k dispozici možnost zaregistrování svých aplikací, ke kterým je umožněna registrace jednotlivých komponent. Při registraci kreditního systému k jedné z aplikací, je pak v kreditním systému vygenerován virtuální účet sloužící prozatím pro příjem poplatků. Každá aplikace, využívající kreditní systém, má tedy vlastní virtuální účet. Nad těmito účty má pak administrátor přehled, co se týče čísla účtu, dále pak aktuálního zůstatku a nakonec také počet blokovaných kreditů.

7 Testování komponenty

Veškerá funkcionalita kreditního systému se v průběhu vývoje testovala na webové aplikaci, jenž měla být taktéž produktem softwarové laboratoře. Nicméně vývoj této aplikace byl v průběhu pozastaven, a to především z důvodu úbytku členů týmu. Webová aplikace je tedy v rozpracovaném stavu, avšak o možnost testování webové služby kreditního systému jsem nebyl ochuzen.

Princip webové aplikace byl postaven na možnostech nabídky či poptávky různých žánrů videí, za jejichž obdržení by měli uživatelé platit. Ačkoliv vývoj systému byl ukončen v jeho počátcích, napojení na webovou službu kreditního systému bylo stále možné.

Způsob, jakým si klientská aplikace naimplementuje webovou službu, záleží už jen na ni. V testovací aplikaci jsem zvolil možnost volání metod webové služby pomocí AJAX technologie. Tento způsob byl zvolen u volání metod getAmount() a getVirtualAccount(), které se nachází v hlavičce každé stránky. Ajaxové volání bylo zvoleno především z toho důvodu, že dotazování na webovou službu trvá i několik sekund, což není uživatelsky přívětivé.

Taktéž bylo otestováno napojení na další 2 systémy, centrální a uživatelský. Ke komunikaci s těmito systémy byla v kreditním systému vytvořena rozhraní CentralAdapter, CentralFacade a UserAdapter viz kapitola 5.4.1. Pomocí těchto rozhraní bylo otestováno především nastavení poplatků, k čemuž má pravomoc jak uživatel s rolí superadmin, tak i admin. Dále je do centrálního systému přenášena kompletní transakční historie nebo informace o virtuálních účtech, které byly vygenerovány při registraci kreditního systému k aplikaci.

8 Závěr

Primárním cílem této práce bylo vyvinout kreditní systém plateb za položky či služby na webu. Tento vývoj probíhal v rámci založené softwarové laboratoře, která byla na začátku složena dohromady z 11 členů, z toho tří mentorů.

Po rozdělení do jednotlivých týmů, byl náš tým složen z 5 členů. Každý člen měl stanoveny vlastní cíle, mezi které patřilo především vytvoření webových komponent a systémů. Postupem času se počet členů mého týmu, z různých důvodů, snížil jen na dva. Bylo tedy potřeba převzít a nadále vyvíjet i ty systémy, které byly pro nás důležité, a to především z důvodu otestování jednotlivých komponent.

Jednalo se o centrální systém a klientskou webovou aplikaci, která byla v mém případě využita k testování webové služby kreditního systému.

Výsledkem této práce je tedy vytvoření základní funkcionality dvou systémů a klientské aplikace, které spolu určitým způsobem komunikují. Byly splněny jednotlivé požadavky, které jsme na začátku vývoje specifikovali.

Kreditní systém tedy umožňuje, mimo jiné, několik typů transakcí, mezikteré patří vklad a výběr prostředků, dále převody mezi účty a provedení plateb v klientských aplikacích, s čímž souvisí taktéž kompletní transakční historie. Součástí systému je také webová služba nebo například správa virtuálních účtů a poplatků definovaných k jednotlivým typům transakcí.

Dalším postupem tvorby kreditního systému bude především dosažení co nejvyšší bezpečnosti, a to především v případě webové služby. Systém bude také rozšířen o notifikační službu, která je aktuálně v rozpracované fázi.

Cílem této služby je informování uživatelů o provedení transakcí případně změně jejich stavů.

Kreditní systém je spolu s uživatelským systémem dále napojen na systém centrální. Počet komponent (systémů) se bude v rámci softwarové laboratoře rozšiřovat a právě centrální systém slouží jako centrální uzel, odkud jsou veškeré komponenty řízeny.

Centrální systém aktuálně disponuje úplnou správou aplikací a komponent včetně registrace komponent k registrovaným aplikacím. V případě kreditního systému je tedy prozatím možné nastavovat poplatky jednotlivých typů transakcí a to jak globálně, tak i pro zvolenou aplikaci. K dispozici je také transakční historie, jejíž obsahem jsou buď transakce určité aplikace nebo úplná transakční historie, ke které má přístup pouze uživatel s rolí superadmin. V rámci napojeného uživatelského systému je pak v centrálním systému zavedena správa uživatelů. A to jak webových uživatelů, tak i administrátorů.

Centrální systém bude dále rozšiřován o další možnosti nastavení jak sebe samotného, tak i jednotlivých komponent. Mezi možnosti dalšího nastavení kreditního systému bych uvedl například správu vytvořených virtuálních účtů, definování omezení například pro neověřené uživatele.

Klientská aplikace, jenž sloužila jako testovací aplikace našich komponent, nyní ještě zcela nesplňuje naše požadavky. Další vývoj této aplikace však prozatím v plánu není.

Centrální, kreditní i uživatelský systém, jsou nyní ve fázi dalšího vývoje, kterému se bude nadále věnovat softwarová laboratoř. Tyto systémy prozatím nepodstoupily

zkouškou většího zatížení klientských požadavků. Je tak tedy pravděpodobné, že tyto systémy budou muset, po nasazení do ostrého provozu, podstoupit optimalizační proces.

9 Reference

[1] PHP: Hypertext Preprocessor. [online]. © 2001 - 2014 [cit. 2014-04-18]. Dostupné z: http://php.net/

[2] PHP – Wikipedie. *Wikipedie, otevřená encyklopedie* [online]. 2014-04-11 [cit. 2014-04-18]. Dostupné z: http://cs.wikipedia.org/wiki/Php

[3] MySQL - Wikipedie. *Wikipedie, otevřená encyklopedie* [online]. 2014-02-08 [cit. 2014-04-18]. Dostupné z: http://cs.wikipedia.org/wiki/MySQL

[4] High Performance PHP Framework for Web Development - Symfony. [online]. [cit. 2014-04-18]. Dostupné z: http://http://symfony.com/

[5] What is Symfony2? . *Fabien Potencier* [online]. 2011-10-25 [cit. 2014-04-18]. Dostupné z: http://fabien.potencier.org/article/49/what-is-symfony2

[6] Homepage - Twig - The flexible, fast, and secure PHP template engine. [online]. © 2010 - 2012 [cit. 2014-04-18]. Dostupné z: http://twig.sensiolabs.org/

[7] Twig Documentation. *Homepage - Twig - The flexible, fast, and secure PHP template engine* [online]. © 2010 - 2012 [cit. 2014-04-20]. Dostupné z: http://twig.sensiolabs.org/documentation

[8] Home - Doctrine Project. [online]. © 2006 - 2014 [cit. 2014-04-18]. Dostupné z: http://www.doctrine-project.org/

[9] Doctrine 2: úvod do systému. *Zdroják — o tvorbě webových stránek a aplikací* [online]. 2014-07-21 [cit. 2014-04-18]. Dostupné z: http://www.zdrojak.cz/clanky/doctrine-2-uvod-do-systemu/

[10] SOAP – Wikipedie. *Wikipedie, otevřená encyklopedie* [online]. 2013-11-29 [cit. 2014-04-18]. Dostupné z: http://cs.wikipedia.org/wiki/SOAP

[11] WS-Security - Wikipedia, the free encyclopedia. *Wikipedia, the free encyclopedia* [online]. 2014-04-14 [cit. 2014-04-18]. Dostupné z: http://en.wikipedia.org/wiki/WS-Security

[12] Web Services Description Language – Wikipedie. *Wikipedie, otevřená encyklopedie* [online]. 2013-06-22 [cit. 2014-04-18]. Dostupné z: http://cs.wikipedia.org/wiki/Web_Services_Description_Language

[13] History - PayPal. *PayPal Press Center - PayPal* [online]. © 1999 - 2014 [cit. 2014-04-18]. Dostupné z: https://www.paypal-media.com/history

[14] PayPal – Wikipedie. *Wikipedie, otevřená encyklopedie* [online]. 2013-10-31 [cit. 2014-04-18]. Dostupné z: http://cs.wikipedia.org/wiki/PayPal

[15] PayPal - Wikipedia, the free encyclopedia. *Wikipedia, the free encyclopedia* [online]. 2014-04-12 [cit. 2014-04-18].
Dostupné z: http://en.wikipedia.org/wiki/PayPal

[16] PayPal Payments Standard Overview. *Home: PayPal Developer* [online]. © 1999 - 2014 [cit. 2014-04-18]. Dostupné z: https://developer.paypal.com/docs/classic/paypal-payments-standard/integration-guide/wp_standard_overview/

[17] Development & Integration Guides. *Home: PayPal Developer* [online]. © 1999 - 2014 [cit. 2014-04-18]. Dostupné z: https://developer.paypal.com/docs/classic/products/#wpp

[18] PayPal Payments Standard and Button Manager: Getting Started. *Home: PayPal Developer* [online]. © 1999 - 2014 [cit. 2014-04-18]. Dostupné z: https://developer.paypal.com/docs/classic/adaptive-payments/gs_AdaptivePayments/

[19] Express Checkout API Getting Started Guide. *Home: PayPal Developer* [online]. © 1999 - 2014 [cit. 2014-04-18]. Dostupné z: https://developer.paypal.com/docs/classic/express-checkout/gs_expresscheckout/

[20] Instant Payment Notification: Getting Started. *Home: PayPal Developer* [online]. © 1999 - 2014 [cit. 2014-04-21]. Dostupné z: https://developer.paypal.com/webapps/developer/docs/classic/ipn/gs_IPN/

[21] Introducing Adaptive Payments. *Home: PayPal Developer* [online]. © 1999 - 2014 [cit. 2014-04-21]. Dostupné z: https://developer.paypal.com/docs/classic/adaptive-payments/integration-guide/APIntro/

[22] About Skrill. *Online Payment System — Send Money, Receive Money — Skrill* [online]. 2011 [cit. 2014-04-21]. Dostupné z: https://www.skrill.com/en/about-us/

[23] Platební metody PayU. *PayU pro e-shopy* [online]. [cit. 2014-04-18]. Dostupné z: http://www.payu.cz/platebni-metody-pro-e-shopy

[24] Poplatky u PayU. *PayU pro e-shopy* [online]. [cit. 2014-04-18]. Dostupné z: http://www.payu.cz/poplatky

[25] O nás – GoPay. *GoPay – Užijte si pohodlí online plateb* [online]. © 2014 [cit. 2014-04-18]. Dostupné z: http://www.gopay.com/cs/o-nas

[26] Základní informace o bráně Klikni a daruj. *PaySec - bezpečné nakupování na internetu* [online]. © 2007 [cit. 2014-04-20]. Dostupné z: http://www.paysec.cz/CmsPage.aspx?Id=nonProfitOrganisation

[27] O produktu / GP webpay - moderní a bezpečná internetová platební brána. [online]. © 2013 [cit. 2014-04-20]. Dostupné z: http://gpwebpay.cz/About

[28] Service Container. *High Performance PHP Framework for Web Development - Symfony* [online]. [cit. 2014-04-22]. Dostupné z: http://symfony.com/doc/current/book/service_container.html

[29] XSD Date and Time Data Types. *HW3Schools Online Web Tutorials* [online]. © 1999 - 2014 [cit. 2014-04-24]. Dostupné z: http://www.w3schools.com/schema/schema_dtypes_date.asp

[30] Cron – Wikipedie. *Wikipedie, otevřená encyklopedie* [online]. 2013-03-19 [cit. 2014-04-23]. Dostupné z: http://cs.wikipedia.org/wiki/Cron

A WSDL

```
<?xml version="1.0"?>
<definitions name="Payment"
             targetNamespace="urn:Payment"
             xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns:tns="urn:Payment"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <types>
    <xsd:schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="urn:Payment
        ">

      <!-- AMOUNT -->
      <xsd:element name="getAmountRequest">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="user" type="string" minOccurs = "1" maxOccurs = "1" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="getAmountResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="balance" type="decimal" minOccurs = "1" maxOccurs = "1" /
                >
            <xsd:element name="blocked" type="decimal" minOccurs = "1" maxOccurs = "1" /
                >
            <xsd:element name="error" type="string" minOccurs = "1" maxOccurs = "1" />
            <xsd:element name="code" type="integer" minOccurs = "0" maxOccurs = "1" />
            <xsd:element name="message" type="string" minOccurs = "0" maxOccurs = "1" /
                >
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>

      <!-- PAYMENT -->
      <xsd:complexType name="transaction">
        <xsd:sequence>
          <xsd:element name="id" type="xsd:integer" minOccurs = "1" maxOccurs = "1"/>
          <xsd:element name="custom" type="xsd:string" minOccurs = "0" maxOccurs = "1"/>
          <xsd:element name="state" type="xsd:string" minOccurs = "1" maxOccurs = "1"/>
          <xsd:element name="transaction_type" type="xsd:string" minOccurs = "1"
              maxOccurs = "1"/>
          <xsd:element name="payer_account" type="xsd:string" minOccurs = "0" maxOccurs
              = "1"/>
          <xsd:element name="receiver_account" type="xsd:string" minOccurs = "0"
              maxOccurs = "1"/>
          <xsd:element name="payer_amount" type="xsd:decimal" minOccurs = "1"
              maxOccurs = "1"/>
          <xsd:element name="receiver_amount" type="xsd:decimal" minOccurs = "1"
              maxOccurs = "1"/>
        </xsd:sequence>
```

```
        </xsd:complexType>
        <xsd:complexType name="proceedPaymentRequest">
            <xsd:sequence>
                <xsd:element name="payer" type="xsd:string" minOccurs = "1" maxOccurs = "1"/>
                <xsd:element name="amount" type="xsd:decimal" minOccurs = "1" maxOccurs = "1"/
                    >
                <xsd:element name="custom" type="xsd:string" minOccurs = "0" maxOccurs = "1"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="proceedPaymentResponse">
            <xsd:sequence>
                <xsd:element name="error" type="boolean" minOccurs = "1" maxOccurs = "1" />
                <xsd:element name="code" type="xsd:integer" minOccurs = "0" maxOccurs = "1"/>
                <xsd:element name="message" type="xsd:string" minOccurs = "0" maxOccurs = "1"/
                    >
                <xsd:element name="transaction" type="tns:transaction" minOccurs = "0"
                    maxOccurs = "1"/>
            </xsd:sequence>
        </xsd:complexType>

        <xsd:complexType name="DateTime">
            <xsd:sequence>
                <xsd:element minOccurs="1" maxOccurs="1" name="date" type="xsd:string"/>
                <xsd:element minOccurs="1" maxOccurs="1" name="timezone_type" type="xsd:int"/
                    >
                <xsd:element minOccurs="1" maxOccurs="1" name="timezone" type="xsd:string"/>
            </xsd:sequence>
        </xsd:complexType>


        <xsd:element name="getVirtualAccountRequest">
            <complexType>
                <sequence>
                    <xsd:element name="user" type="string" minOccurs = "1" maxOccurs = "1" />
                </sequence>
            </complexType>
        </xsd:element>
        <xsd:complexType name="getVirtualAccountResponse">
            <xsd:sequence>
                <xsd:element name="error" type="boolean" minOccurs = "1" maxOccurs = "1" />
                <xsd:element name="code" type="xsd:integer" minOccurs = "0" maxOccurs = "1"/>
                <xsd:element name="message" type="xsd:string" minOccurs = "0" maxOccurs = "1"/
                    >
                <xsd:element name="accountNumber" type="xsd:string" minOccurs = "0"
                    maxOccurs = "1"/>
            </xsd:sequence>
        </xsd:complexType>

    </xsd:schema>
</types>
<message name="getAmountRequest">
    <part name="request" element="tns:getAmountRequest"/>
</message>
<message name="getAmountResponse">
```

```
    <part name=”response” element=”xsd:getAmountResponse” />
</message>

<message name=”proceedPaymentRequest”>
    <part name=”parameters” type=”tns:proceedPaymentRequest” />
</message>
<message name=”proceedPaymentResponse”>
    <part name=”return” type=”tns:proceedPaymentResponse” />
</message>

<message name=”getVirtualAccountRequest”>
    <part name=”request” element=”tns:getVirtualAccountRequest”/>
</message>
<message name=”getUserVirtualAccountResponse”>
    <part name=”response” element=”xsd:getVirtualAccountResponse” />
</message>

<portType name=”WSPaymentPort”>
    <operation name=”getAmount”>
        <input message=”tns:getAmountRequest” />
        <output message=”tns:getAmountResponse” />
    </operation>
    <operation name=”proceedAppPayment”>
        <input message=”tns:proceedPaymentRequest” />
        <output message=”tns:proceedPaymentResponse” />
    </operation>
    <operation name=”getVirtualAccount”>
        <input message=”tns:getVirtualAccountRequest” />
        <output message=”tns:getVirtualAccountResponse” />
    </operation>
</portType>
<binding name=”WSPaymentBinding” type=”tns:WSPaymentPort”>
    <soap:binding style=”rpc” transport=”http: // schemas.xmlsoap.org/soap/http” />
    <operation name=”getAmount”>
        <soap:operation soapAction=”urn:getAmountAction” />
        <input>
            <soap:body use=”encoded” namespace=”urn:Payment” encodingStyle=”http://schemas.
                xmlsoap.org/soap/encoding/” />
        </input>
        <output>
            <soap:body use=”encoded” namespace=”urn:Payment” encodingStyle=”http://schemas.
                xmlsoap.org/soap/encoding/” />
        </output>
    </operation>
    <operation name=”getVirtualAccount”>
        <soap:operation soapAction=”urn:getVirtualAccountAction” />
        <input>
            <soap:body use=”encoded” namespace=”urn:Payment” encodingStyle=”http://schemas.
                xmlsoap.org/soap/encoding/” />
        </input>
        <output>
            <soap:body use=”encoded” namespace=”urn:Payment” encodingStyle=”http://schemas.
                xmlsoap.org/soap/encoding/” />
        </output>
```

```
        </operation>
        <operation name="proceedAppPayment">
            <soap:operation soapAction="urn:PaymentAction" />
            <input>
                <soap:body use="encoded" namespace="urn:Payment" encodingStyle="http://schemas.
                    xmlsoap.org/soap/encoding/" />
            </input>
            <output>
                <soap:body use="encoded" namespace="urn:Payment" encodingStyle="http://schemas.
                    xmlsoap.org/soap/encoding/" />
            </output>
        </operation>
    </binding>

    <service name="WSPaymentService">
        <port name="WSPaymentPort" binding="tns:WSPaymentBinding">
            <soap:address location="http://[domena]/wsdl" />
        </port>
    </service>

</definitions>
```

B Třídní diagram

Z důvodu větší rozsáhlosti třídního diagramu, je tento dokument založen v kapse této práce.

C Sekvenční diagramy

V první části této kapitoly jsou jednotlivé diagramy popsány, v druhé částí následně znázorněny.

SD2: Příprava PayPal transakce Jedná se o diagram vyobrazující přípravu PayPal transakce k čemuž jsou využity především instance tříd PaypalTransactions, PaypalTransactionsRepository, Transaction PaypalAccount a PaypalAccountsRepository. Tento sekvenční diagram je vyvolán metodou "prepareDepositTransaction()"a výstupem je pak entita hlavní (rodičovské) transakce.

U každé PayPal transakce je v databázi uchováno ID transakce, kterém bylo zasláno systémem PayPal. Toto ID PayPal zasílá ve svých IPN zprávách a v kreditním systému je pak využito pro párovaní transakcí. Body (2-10) jsou vykonány pouze v případě, že nebyla nalezena žádná transakce s daným ID transakce. Je tedy potřeba vytvořit transakci novou. Pokud byla nalezena alespoň jedna transakce, je proveden bod (11).

(1) Vyhledání všech PayPal transakcí podle ID transakce, které bylo získáno z IPN zprávy (txn_id).

(2) Metoda "verifyEncodedString()"ověří token uživatele, čímž dochází k ujištění, že příjemce platby nebyl nikým změněn.

(3) Dohledání entity PayPal účtu příjemce.

(4) Pokud nebyl PayPal účet příjemce nalezen, vytvoří se nový.

(5) Dohledání entity PayPal účtu plátce.

(6) Pokud nebyl PayPal účet plátce nalezen, vytvoří se nový.

(7) V odděleném diagramu SD4 (viz obrázek 10) se provede nastavení (atributů) nově vytvářené transakce.

(8) Nastavená transakce se uloží do paměti.

(9) Pomocí odděleného diagram (viz 11 je vyobrazena příprava PayPal poplatku.

(10) Transakce připravené z bodu (9) se uloží do paměti.

(11) V odděleném diagramu SD6 (viz obrázek 12) se provede alternativní blok.

SD3: Nový PayPal účet Jedná se o diagram znázorňující vytvoření nového PayPal účtu v kreditním systému k čemuž jsou využity především instance tříd PaypalAccount, User, UserRepository, Currency, CurrencyReposiory, PaypalAccountsReposiroty. Tento sekvenční diagram je vyvolán metodou "savePAccount()"a výstupem je pak entita vytvořeného PayPal účtu.

(1) Dohledání entity uživatele.

(2) Dohledání entity potřebné měny, která se páruje podle kódu měny získané z IPN zprávy.

(3) Pokud taková měna neexistuje, vytvoří se nová. Bude ji však nastaven příznak "hidden", díky kterému se nebude v kreditním systému zobrazovat.

(4) Pomocí získaných entit a dat obdržených v IPN zprávě se vytvoří nová entita definující nový PayPal účet.

SD4: Nastavení entity PaypalTransactions Diagram popisuje nastavení entit nové PayPal transakce. Jedná se o entity Transactions a PaypalTransactions. k čemuž jsou využity především instance tříd PaypalTransaction, TransactionType, TransactionTypesRepository, State, StatesRepository, PaypalTransactionsRepository. Tento sekvenční diagram slouží jako rozšíření diagramu SD1 viz obrázek 7. Není tedy vyvolán žádnou operací.

(1) Nastaví se všechny potřebné vlastnosti rodičovské entity (Transactions). Jedná se o účet příjemce, účet plátce, částka příjemce, částka plátce, typ transakce, stav transakce, datum vytvoření transakce.

(2) Pokud je PayPal transakce ve stavu "completed", je nastaveno i datum potvrzení transakce.

(3) Nastavení Paypal transakce (PaypalTransactions) spolu s vazbou na výše nastavenou rodičovskou entitu (Transactions).

SD5: Příprava transakce Paypal poplatku Diagram popisuje nastavení entit určujících PayPal poplatek. Stejně jako u předcházejícího popisu se jedná o entity Transactions a PaypalTransactions. Tato operace zprostředkovává metoda "preparePaypalFeeTransaction()", která je volána až po nastavení hlavní PayPal transakce. Důvodem je nutnost nastavení nadřazené transakce poplatku, kterým je právě hlavní PayPal transakce. Výstupem této operace je právě entita poplatku.

(1) Kromě nastavení ostatních vlastností entity jsem bodem (1) vyzdvihl způsob vytvoření vazby na nadřazenou (rodičovskou) entitu. Tato entita je funkci předána parametrem ($parentTransaction).

SD6: Provedení alternativního bloku Alternativní blok je proveden v případě nalezení alespoň jedné transakce (podtransakce) s daným ID. Účelem tohoto alternativního bloku je změna stavů vyhledaných transakcí a v případě dokončené transakce nastavení data dokončení. Alternativní blok slouží jako rozšíření diagramu SD2 viz obrázek 8, který bylo z důvodu nedostatku prostoru potřeba oddělit. Mezi objekty komunikující v alternativním bloku patří PaypalTransaction, State, PaypalTransactions a Transaction.

(1) Dochází k porovnání aktuálního stavu transakce s novým stavem, který byl poslán systémem PayPal v IPN zprávě.

(2) Pokud jsou tyto stavy odlišné, nastaví se stav nový.

(3) Pokud je transakce ve stavu dokončeno (completed), je nastaveno datum potvrzení transakce.

(4) Po úspěšném dokončení je transakce uložena do paměti.

SD7: Příprava virtuální transakce Příprava virtuální transakce je vyvolána metodou "prepareDepositTransaction()"a to pouze v případě, kdy je transakce zaslaná IPN zprávou ve stavu "completed". Až v tuto chvíli je možné uživateli kreditního systému připsat kredity na jeho primární virtuální účet. V procesu přípravy virtuální transakce mezi sebou komunikují instance tříd VirtualTransaction, Transactions, Transaction, VirtualAccount, PaypalAccount, Rate, TransactionType a State.

(1) Nastavení jednotlivých vlastností entity.

(2) Ověření, zda-li měna, ve které je provedena transakce na straně systému PayPal, je totožná s měnou virtuálního účtu příjemce. Pokud jsou měny odlišné, částka příjemce virtuální transakce je převeda podle aktuálního kurzu ČNB.

(3) Nově vytvořená virtuální transakce se uloží do paměti.

(4) Příprava poplatků viz diagram SD8 14

SD8: Příprava poplatků virtuální transakce Diagram příprava poplatků virtuální transakce znázorňuje komunikaci mezi instancemi tříd VirtualTransaction, TransactionType, Rate, State a Transaction. Operace vytvoří entity uplatněných poplatků a připraví je tak k persistenci.

(1) Metoda "getAssignedFees()"získá všechny poplatky, které mohou být k dané transakci uplatněny.

(2) Cílem metody "prepareFeeTransactions()"je příprava entit uplatněných poplatků, které následně uloží do paměti.

(3) Cyklickým procházením uplatněných poplatků se vytvoří instance entity Transactions s následným nastavením setterů.

(4) V případě, že se měna plátce liší od měny příjemce, dojde k převodu podle aktuálního kurzu ČNB.

(5) Uložení připraveného poplatku do paměti.

SD9: Změna stavů rodičovský transakcí Diagram znázorňuje operaci změny stavů rodičovských transakcí v případě změny stavu potomka. Komunikace probíhá mezi instancemi tříd PaypalTransaction, PaypalTransactionsRepository, State, PaypalTransactions a Transaction. Po provedení změny stavů jsou transakce předpřipraveny k persistenci a dočasně jsou tak uchovány v paměti.

(1) Vyhledání transakcí která jsou systémem PayPal označeny jako rodičovské.

(2) Změna stavů probíhá pouze v případě, že byla nalezena alespoň jedna taková transakce.

(3) V cyklu se pak mění stavy nalezených transakcí, které se následně uloží do paměti a jsou tak připraveny k persistenci do databáze.

SD10: Výběr - požadavek Jedná se o diagram znázorňující vytvoření požadavku na výběr kreditů. V tomto procesu mezi sebou komunikují instance tříd PaypalTransactionsController, PaypalTransaction, VirtualAccount, Rate, VirtualAccountsRepository, Transaction, PaypalAccount, PaypalAPICaller a PaypalAccountsRepository.

(1) Před provedením operace dojde k ověření, zda-li virtuální účet, ze kterého budou kredity odeslány, je účet uživatele provádějící tuto operaci. Návratovou hodnout je pak účet plátce. V případě podvrženého účtu je vyhozena vyjímka.

(2) Pokud je výše uvedená podmínka splněna, je zavolána metoda "prepareCashoutTransaction()", jejimž cílem je provedení požadavku na výběr kreditů.

(3) Zjištění zůstatku na virtuálním účtu.

(4) Ověření dostatečného zůstatku k provedení výběru.

(5) Pokud je měna virtuálního účtu odlišná od měny cílového PayPal účtu, kredity se převedou právě na měnu účtu PayPal podle aktuálního kurzu ČNB.

(6) Metoda "sendPayApiRequest()"zajistí odeslání požadavku na API systému PayPal, který vrátí platební klíč (PayKey) k provedení platby.

(7) Vytvoření a nastavení potřebných entit (Transactions, PaypalTransactions). Z důvodu ušetření prostoru je v diagramu zakreslena pouze jedna zástupná metoda "set*($param), kde symbol "*"zastupuje jednolivé metody (settery). Na instanci entity Transactions se nastavují atributy datum vytvoření (created), typ transakce (transaction_type), stav transakce (state), účet plátce a účet příjemce (sender_account, receiver_account), částka plátce a částka příjemce (sender_amount, receiver_amount).

(8) Na entity PaypalTransactions se pak nastaví vazba na již nastavenou entitu Transactions a klíč platby získaný metodou "sendPayApiRequest()".

(9) Transakce se pomocí metody "addTransaction()"uloží do paměti a připraví se tak k persistenci do databáze.

(10) Příprava virtuální transakce znázorněná v diagramu SD11 (viz obrázek 17).

(11) Uložení již připravených transakcí.

SD11: Příprava virtuální transakce Diagram vztahující se k výše popsané operaci k provedení požadavku na výběr. Tuto operaci rozšiřuje o přípravu virtuální transakce. Dochází ke komunikaci mezi instancemi tříd PaypalTransactionsController, VirtualTransaction a Transaction. Opearce je vyvolána metodou "prepareCashoutTransaction()"volaná na objektu třídy VirtualTransaction.

(1) Nastavení entit virtuální transakce. Jedná se o entity Transactions a VirtualTransactions. V diagramu je opět zakreslen pouze zástupce všech metod (setterů) "set*($parameter)".

(2) Přidání transakce do paměti, čímž je připravena k persistenci do databáze.

(3) Reference na diagram SD8 (viz obrázek 14), který znázorňuje přípravu poplatků.

SD13: Převod kreditů Operace znázorněná sekvenčním diagramem SD13 (viz 18). Mezi objekty, které jsou pro tuto operaci využity patří instance tříd VirtualTransactionsController, VirtualAccount, AccountAdapter, VirtualAccountsRepository, VirtualTransaction a Transaction. Operace je vyvolána potvrzení formuláře. Vstupní data jsou tedy zaslána HTTP metodou POST. Účelem diagramu je jak již název diagramu napovídá, provedení převodu kreditů na uživatelem zvolený virtuální účet. Může se jednat o účet vlastní nebo účet jiného uživatele.

(1) Před provedením operace dojde k ověření, zda-li virtuální účet, ze kterého budou kredity odeslány, je účet uživatele provádějící tuto operaci. Návratovou hodnout je pak účet plátce. V případě podvrženého účtu je vyhozena vyjímka.

(2) Pokud je v POST proměnné "target"uvedena hodnota "1", jedná se o převod na konkrétní virtuální účet. V tomto případě tedy dojde k vyhledání virtuálního účtu podle čísla účtu, které uživatel zadá jako vstup ve formuláři.

(3) V opačném případě se jedná o převod s tím rozdílem, že se na vstupu očekává e-mail příjemce převodu namísto čísla účtu. Podle tohoto e-mailu, systém dohlédá entitu uživatele, k čemuž je však využita uživatelská komponenta, ke které přistupuji přes rozhraní "AccountAdapter". Entita je dále využita pro vyhledání primárního virtuálního účtu uživatele metodou "getPrimaryVAccount()".

(4) Pokud jsou účty plátce i příjemce aktivní, dojde k přípravě převodu viz diagram SD14 19 a následné persistenci.

SD14: Příprava převodu Diagram (viz 19 znázorňuje přípravu entit k persistenci transakce převodu. Komunikace dochází mezi instancemi tříd VirtualTransaction, VirtualAccount a Rate. Operace je vyvolána metodou "prepareTransferTransaction()", jejimž účelem je kromě přípravy entit samotné transakce i příprava poplatků.

(1) Výpočet aktuálního zůstatku daného virtuálního účtu.

(2) Pokud zůstatek není dostatečný nebo se účet plátce shoduje s příjemce, je vyhozena vyjímka a proces končí.

(3) Dochází k porovnání měn. V případě odlišných měn, je částka příjemce převedena podle aktuálního kurzu ČNB na jim stanovenou měnu.

(4) Nastavení entity pomocí "set"metod, které jsou opět zastoupeny jedninou metodou set*($parameter).

(5) Reference na diagram SD8 (viz obrázek 14), znázorňující přípravu poplatků.

SD15: Platba aplikaci Proces, znázorněný diagramem SD15 (viz obrázek 20 je operace webové služby. Je tedy vyvolána SOAP požadavkem, který je zpracován rozhraním "WSPaymentFacade". K provedení této operace jsou dále využity instance tříd

UserAdapter, VirtualTransaction, User, Transaction, UsersRepository, AccountApplication, Application, CentralAdapter, ApplicationsRepository a VirtualAccount. Výstupem je objekt vytvořené platební transakce.

(1) Získání entity uživatele z uživatelské komponenty, ke které přistupuji přes rozhraní "AccountAdapter".

(2) Podle získané entity se vyhledá entita uživatele kreditního systému, který zastává roli plátce.

(3) Vyhledání aplikace v centrálním systému, ke kterému přistupuji pomocí rozhraní "CentralAdapter".

(4) Ze získané entity aplikace lze vyčíst uživatele (role: admin), na kterého je aplikace zaregistrována. Podle tohodo uživatele je pak pomocí metody "getUserByComponent()"vyhledána entita uživatele zastávající roli příjemce.

(5) Metoda "getPaymentApplication()"přemapuje entitu aplikace centrálního systému na entitu kreditního systému, se kterou se nadále pracuje.

(6) Vyhledání virtuálního účtu plátce.

(7) Vyhledání virtuálního účtu příjemce.

(8) Zjištění zůstatku na virtuálním účtu plátce.

(9) Pokud je dostatečný zůstatek k provedení transakce, dojde k připravení transakce.

(10) Uložení transakce do paměti.

(11) Reference na sekvenční diagram SD8 14, znázorňující přípravu poplatků.

(12) Pesistence všech připravených transakcí.

(13) Odeslání e-mailu, který webového uživatele informuje o platbě a zároveň jej vyzývá k potvrzení transakce.

SD16: Vytvoření virtuálního účtu Jedná se o operaci vyvolanou potvrzením HTML formuláře, přičemž data jsou zaslána HTTP metodou POST. Kompletní diagram je znázorněn v sekvenčním diagramu SD16 (viz 21).

(1) Vyhledání entity přihlášeného uživatele.

(2) Vyhledání entity uživatelem vybrané měny.

(3) Nastavení entity Accounts, kde byly jednotlivé "set"metody (settery) zastoupeny jedinou metodou "set*($parameter)".

(4) Blok spouštějící se v případě, kdy uživatel zvolil možnost vygenerování náhodného čísla účtu.

(5) Pokud uživatel zadal vlastní číslo účtu, dojde k ověření, zda-li je číslo unikátní.

(6) Vyhledání všech účtů uživatele.

(7) Pokud nebyl nalezen žádný účet, nově vytvářený účet je označen jako primární.

(8) Persistence virtuálního účtu.

(9) Přidělení účtu aplikacím. Tato operace je znázorněna v sekvenčním diagramu SD17 (viz 22)

SD17: Přidělení VÚ k aplikacím Operace vyvolaná metodou ”assignApplication()”využívá instancí tříd AccountApplication, VirtualAccount, VirtualAccountsRepository, AccountsApplicationsRepository, Application a ApplicationRepository. Návratovou hodnotou této operace je pak id vytvořené vazby mezi účtem a aplikací.

(1) Vyhledání entity účtu podle ID.

(2) Vyhledání entity aplikace podle ID.

(3) Nastavení entity ”AccountsApplications”.

(4) Persistence entity.

SD18: Aktivace/deaktivace VÚ Diagram znázorňující aktivaci a deaktivaci virtuálního účtu je uveden na obrázku 23. Operace je vyvolána kliknutím na odkaz ”aktivace”případně ”deaktivace”náležící danému účtu.

(1) Ověření hash řetězce předaným HTTP metodou GET.

(2) Metoda ”verifyAccount()”ověří, zda-li s účtem manipuluje skutečný vlastník.

(3) Vyhledání entity typu ”VirtualAccounts”.

(4) Běh operace pokračuje v případě, že se nejedná o primární účet (atribut ”main”).

(5) Pokud se jedná o aktivaci účtu, nastaví se atribut ”active”na hodnotu TRUE. V opačném případě se danému účtu odeberou přiřazené aplikace, což zařídí metoda ”unassignApplication()”a nastaví set atribut ”active”na hodnotu FALSE.

(6) Persistence entity.

SD19: Změna primárního VÚ Proces změny primárního virtuálního účtu je znázorněn v diagramu SD19 (viz 24). Operace je vyvolána metodou ”switchPrimaryVirtualAccount()”přičemž požadavek je nejdříve zpracová controllerem, který před spuštěním provede určitá ověření. kde dochází ke komunikaci mezi instancemi tříd VirtualAccountsController, VirtualAccount a VirtualAccountsRepository.

(1) Ověření hash řetězce předaným HTTP metodou GET.

(2) Metoda ”verifyAccount()”ověří, zda-li s účtem manipuluje skutečný vlastník.

(3) Vyhledání entity aktuálního primáního účtu.

(4) Vyhledání entity nově zvoleného primárního účtu.

(5) Pokud účty nejsou stejné, dojde k záměně primárních účtů.

SD20: Odebrání VÚ Operace odebrání virtuálního účtu je znázorněna v diagramu SD20 (viz 25). Operace je vyvolána HTTP požadavkem, který je zpracován controllerem (”VirtualAccountsController”). Mezi objekty, které jsou při komunikaci využity patří také instance třídy VirtualAccount, VirtualAccountsRepository, VirtualAccounts a Accounts. Výstupem je pak přesměrování na určitou stránky a zobrazení zprávy o úspěšném či neúspěšném provedení operace.

(1) Ověření hash řetězce předaným HTTP metodou GET.

(2) Metoda ”verifyAccount()”ověří, zda-li s účtem manipuluje skutečný vlastník.

(3) Zjištění aktuálního zůstatku na virtuálnním účtu.

(4) V případě nenulového zůstatku je proces ukončen.

(5) Metodou ”deleteVirtualAccount()”je vyvolán proces mazání, kde se nejdříve získá entita virtuálního účtu. Pomocí entity se ověří, zda-li se nejedná o primární účet, který není možné smazat a následně se účet smaže. Jak je možné vidět, nedochází k fyzickému smazání záznamu z databáze, ale pouze o nastavení atributu ”deleted”na aktuální datum.

(6) Persistence entity.

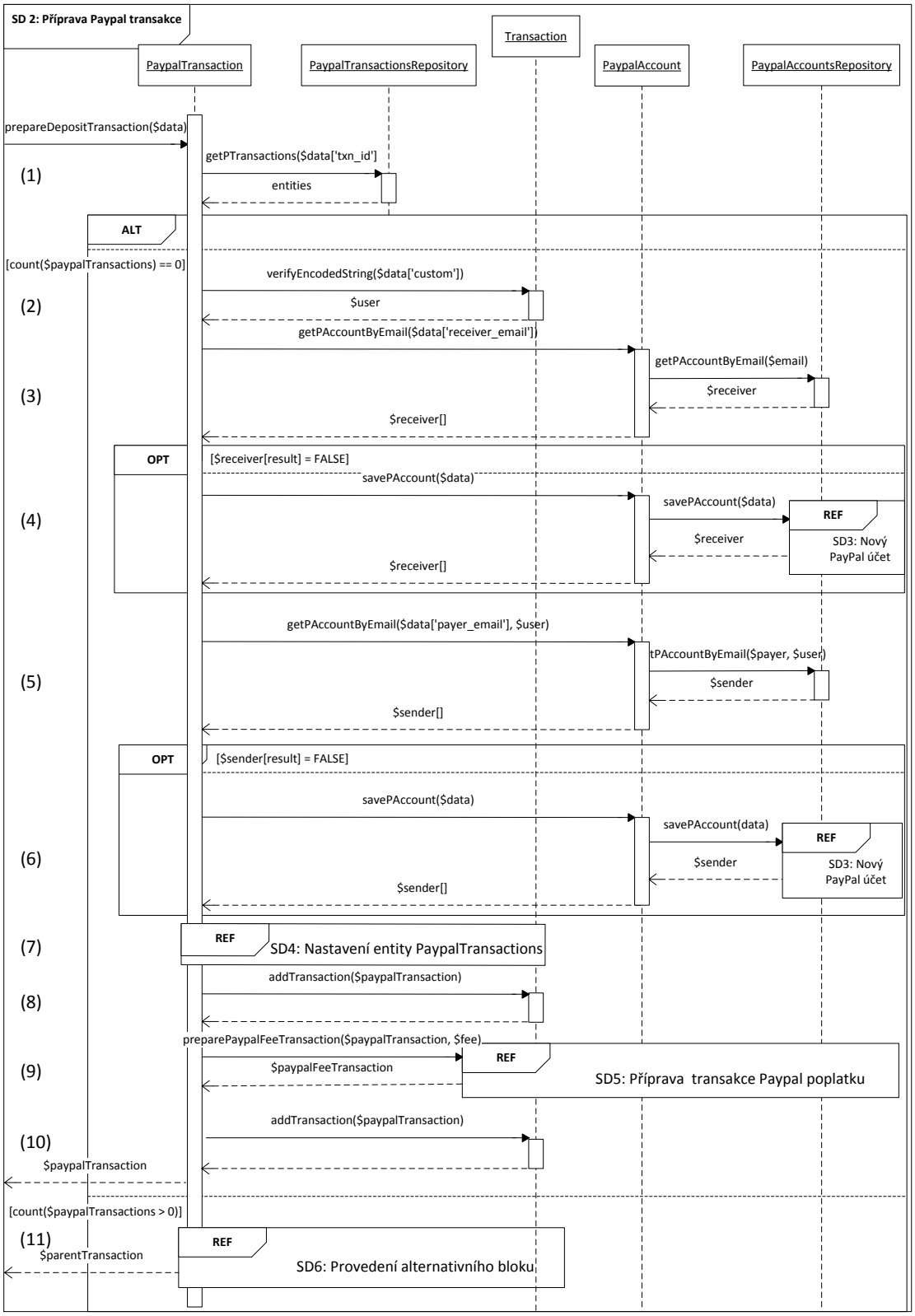
SD21: Správa přidělování VÚ k aplikacím Operace je znázorněna diagramem SD21 (viz 26) a je vyvolána metodou ”assignApplicationAccounts()”. Vstupem jsou dane z formuláře, konkrétně z formulářového prvku multiselect. Účelem je tedy virtuální účet buď přidělit nebo odebrat zvoleným aplikacím.

(1) Metoda ”verifyAccount()”ověří, zda-li s účtem manipuluje skutečný vlastník.

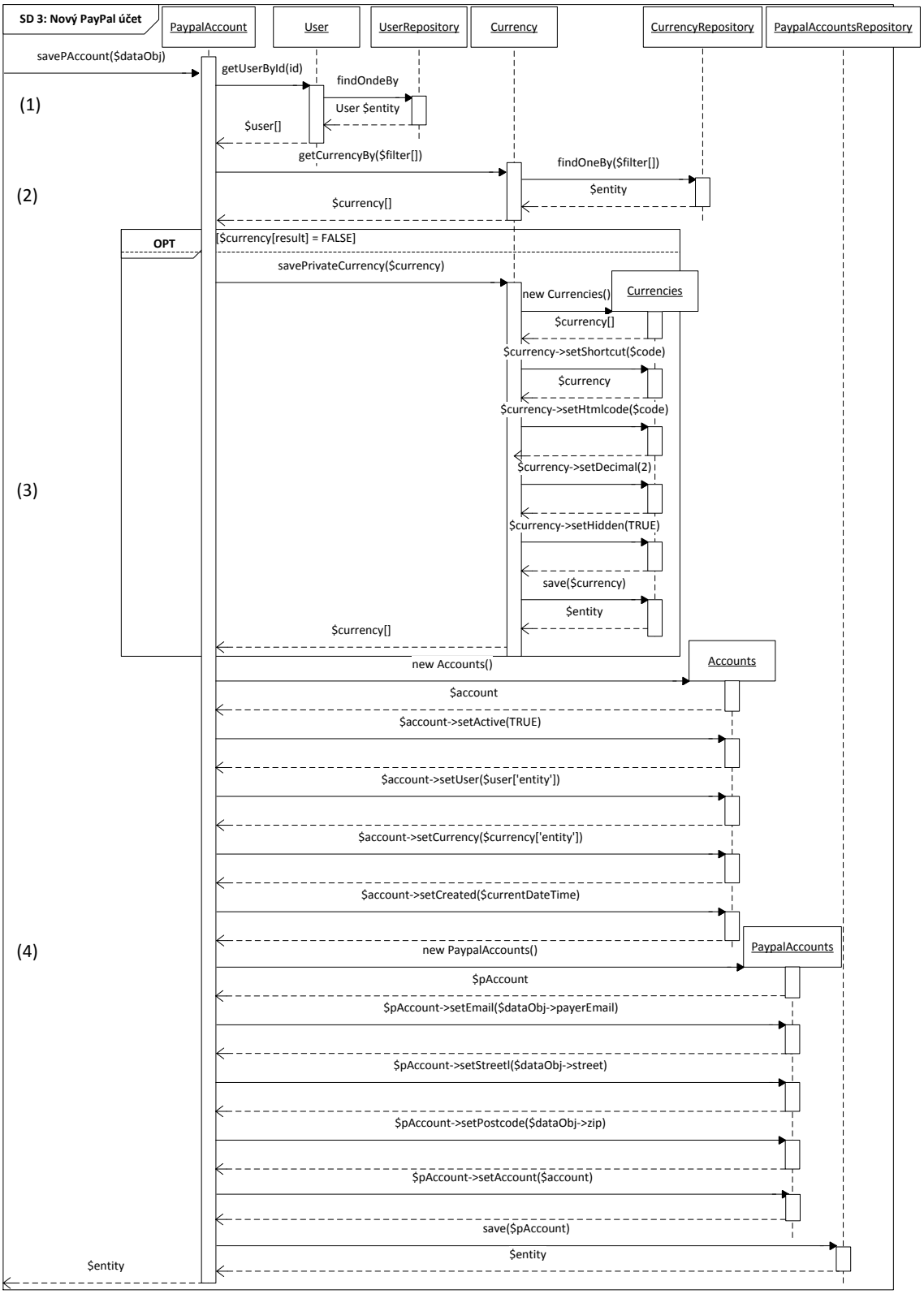
(2) Pokud je účet aktivní jsou pomocí privátní metody zjištěny provedené změny v přiřazení. V opačném případě je vyhozena informativní hláška o neprovedené operaci.

(3) Cyklicky se projde pole aplikací k odebrání a pomocí metody ”unassignApplication()”se odebrání provede.

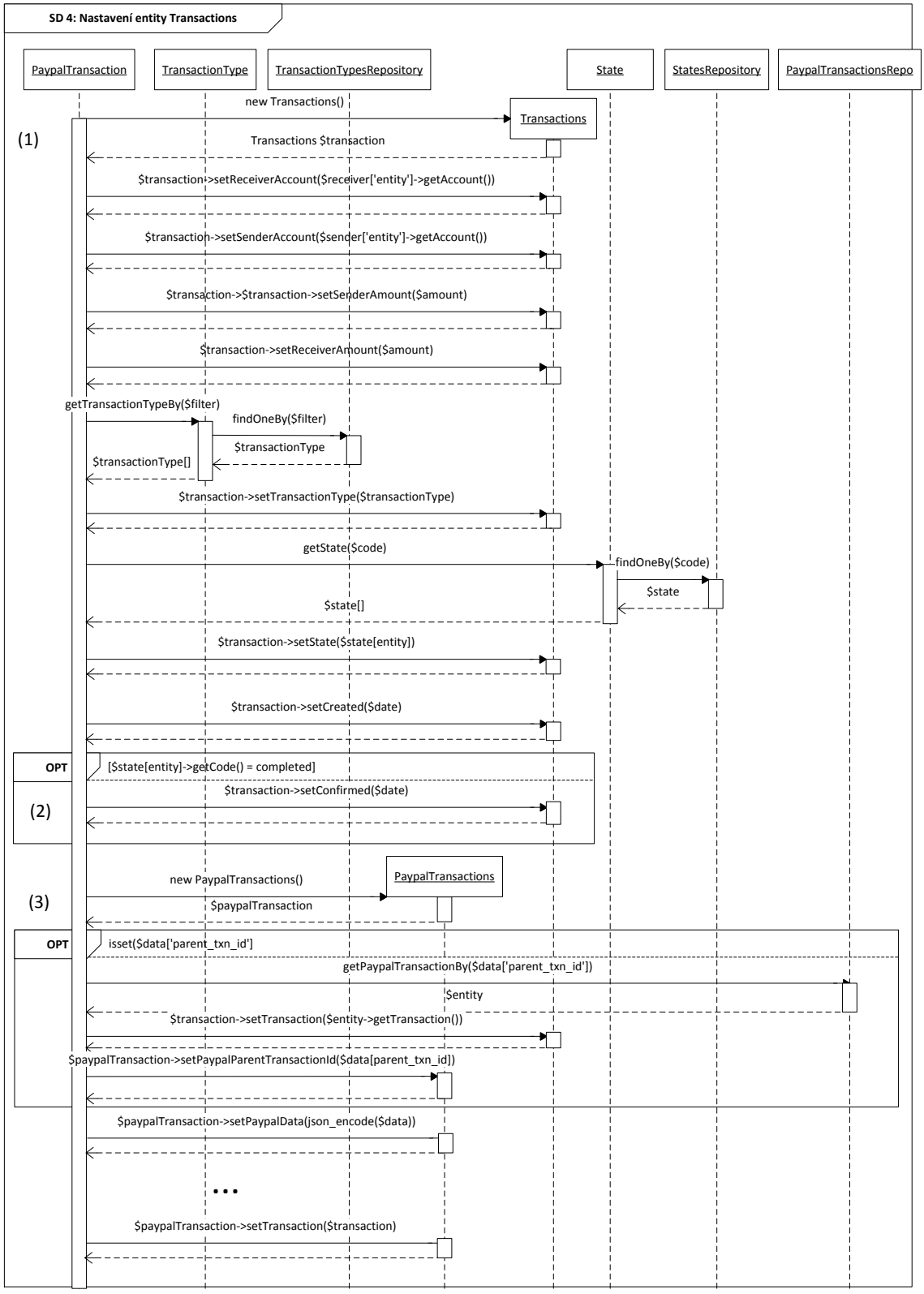
(4) Cyklicky se projde pole aplikací, kde se před samotným přidělení provede ujištění, zda-li aplikace není přidělena již k jinému uživatelova účtu. Pokud k jinému účtu přidělená je, kreditní systém se pokusí o její odebrání a přidělí k nově zvolenému, což je znázorněno na diagramu SD17 viz obrázek 22.



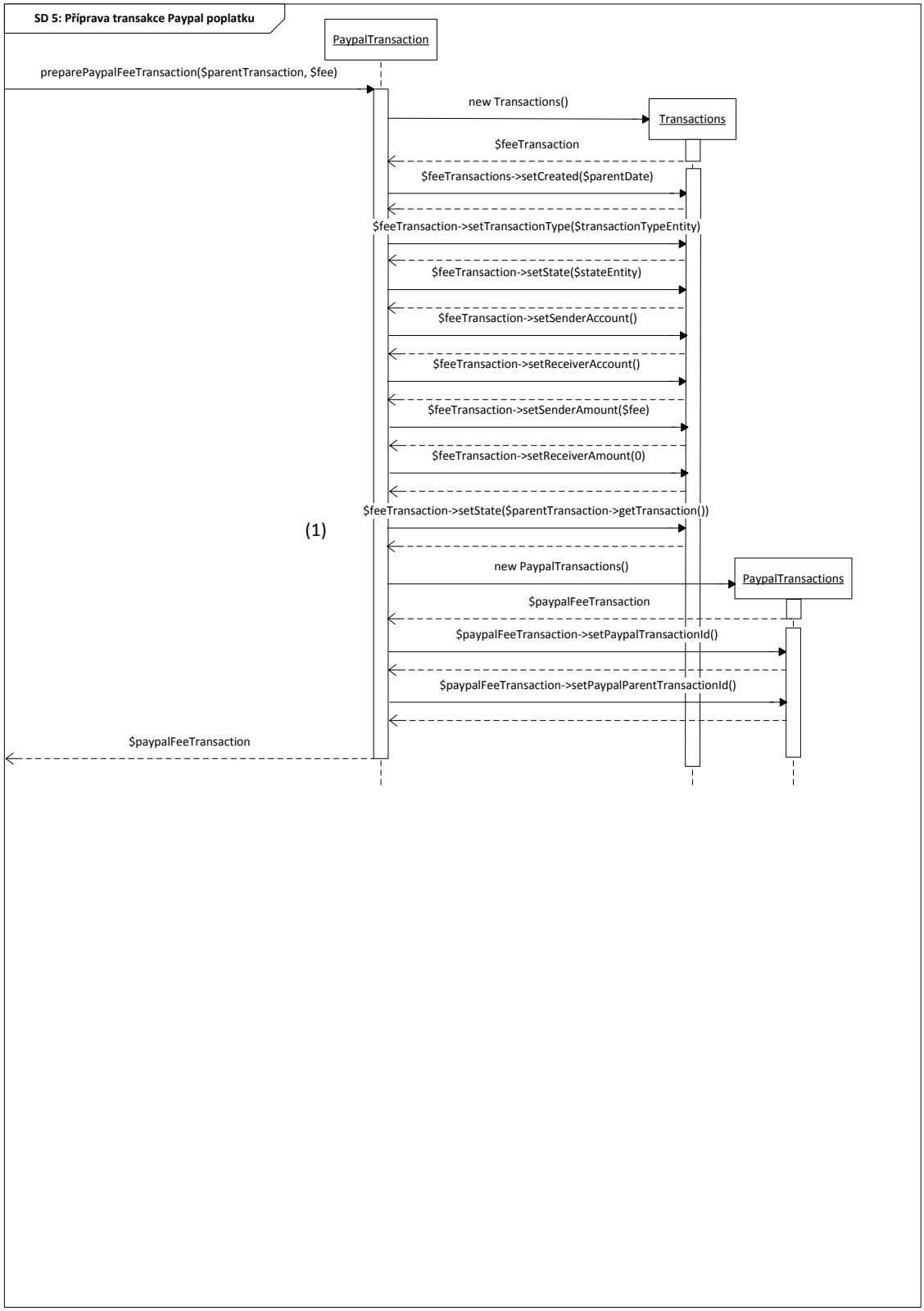
Obrázek 8: SD2: Příprava Paypal transakce



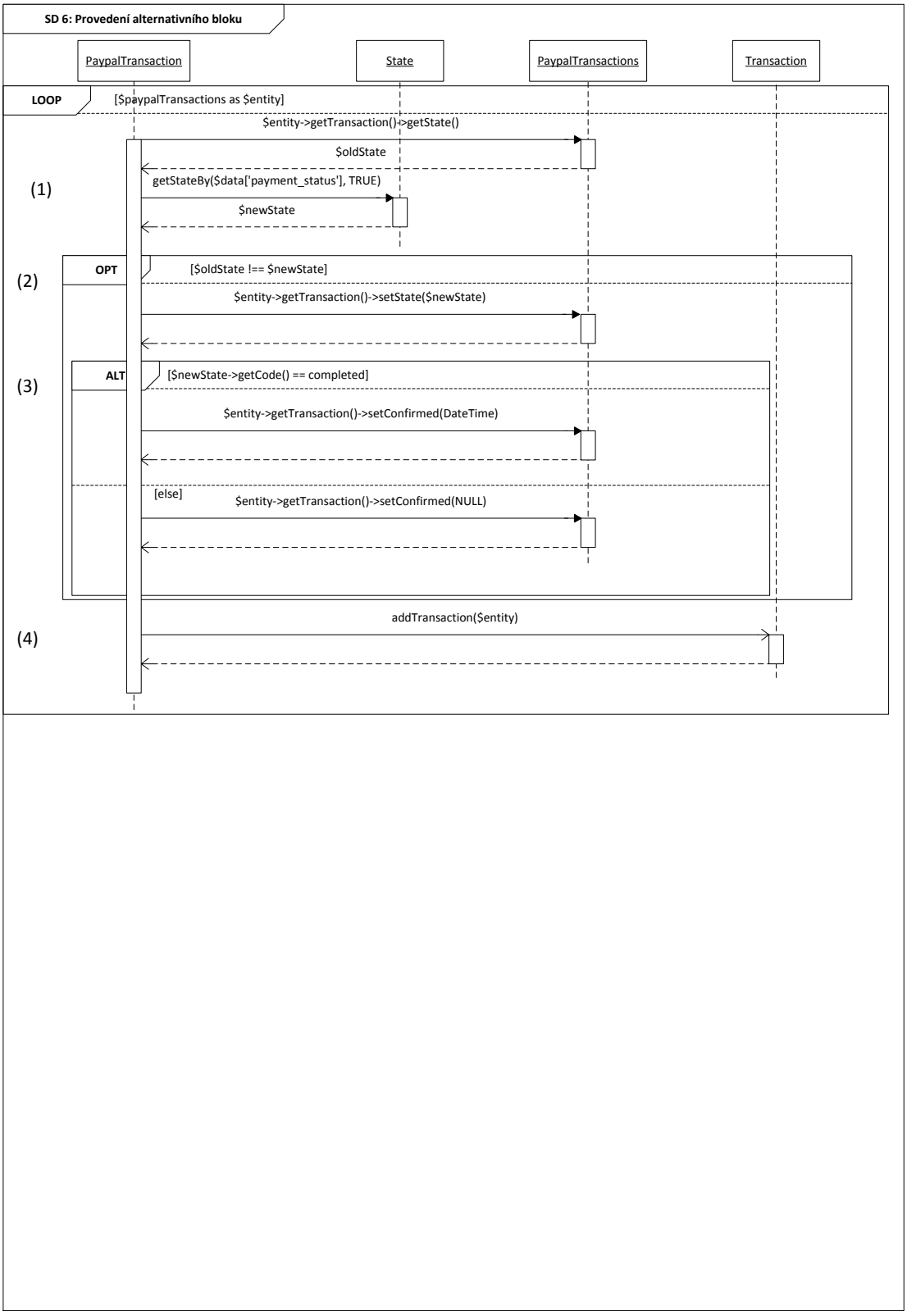
Obrázek 9: SD3: Nový PayPal účet



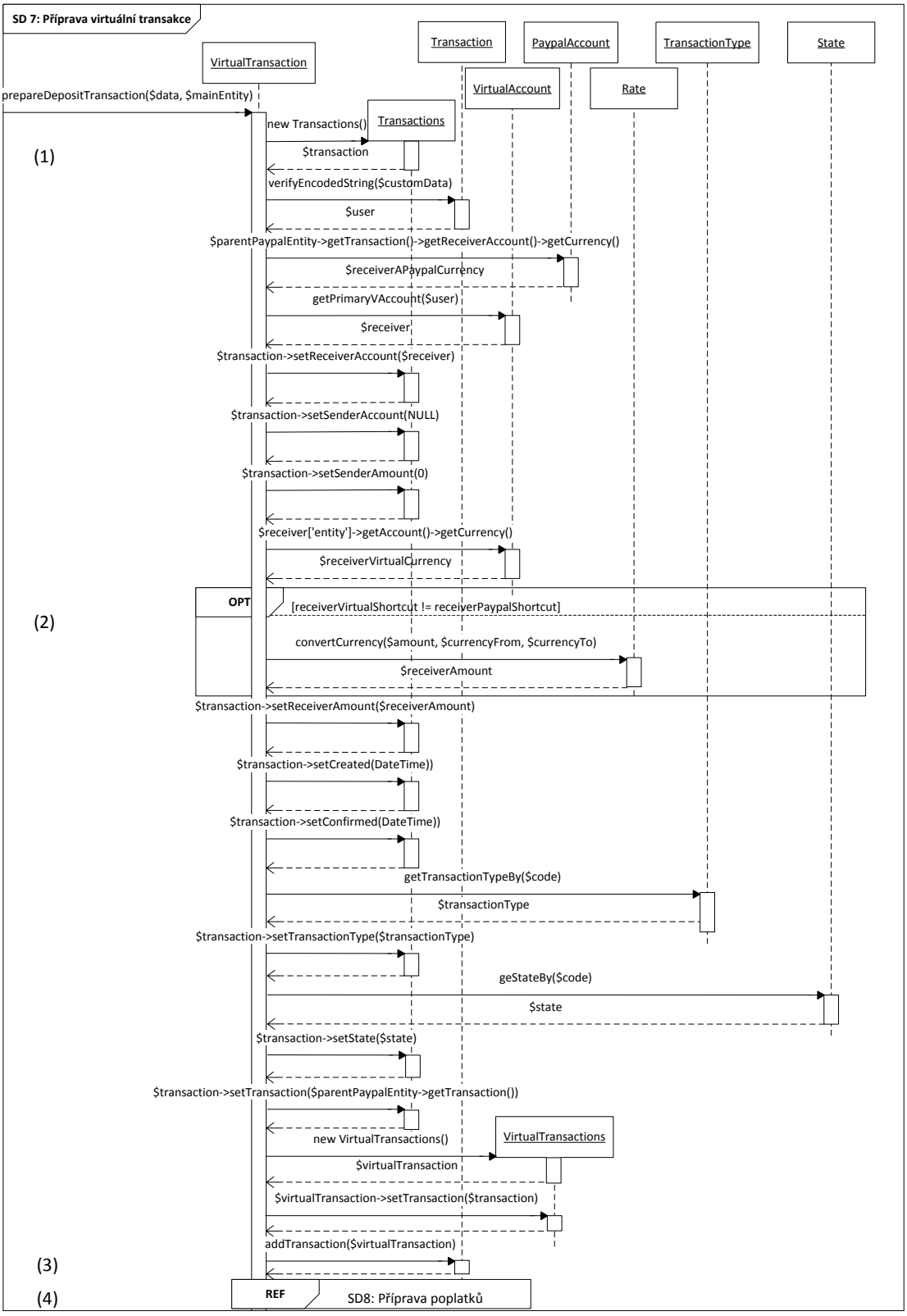
Obrázek 10: SD4: Nastavení entity Transactions



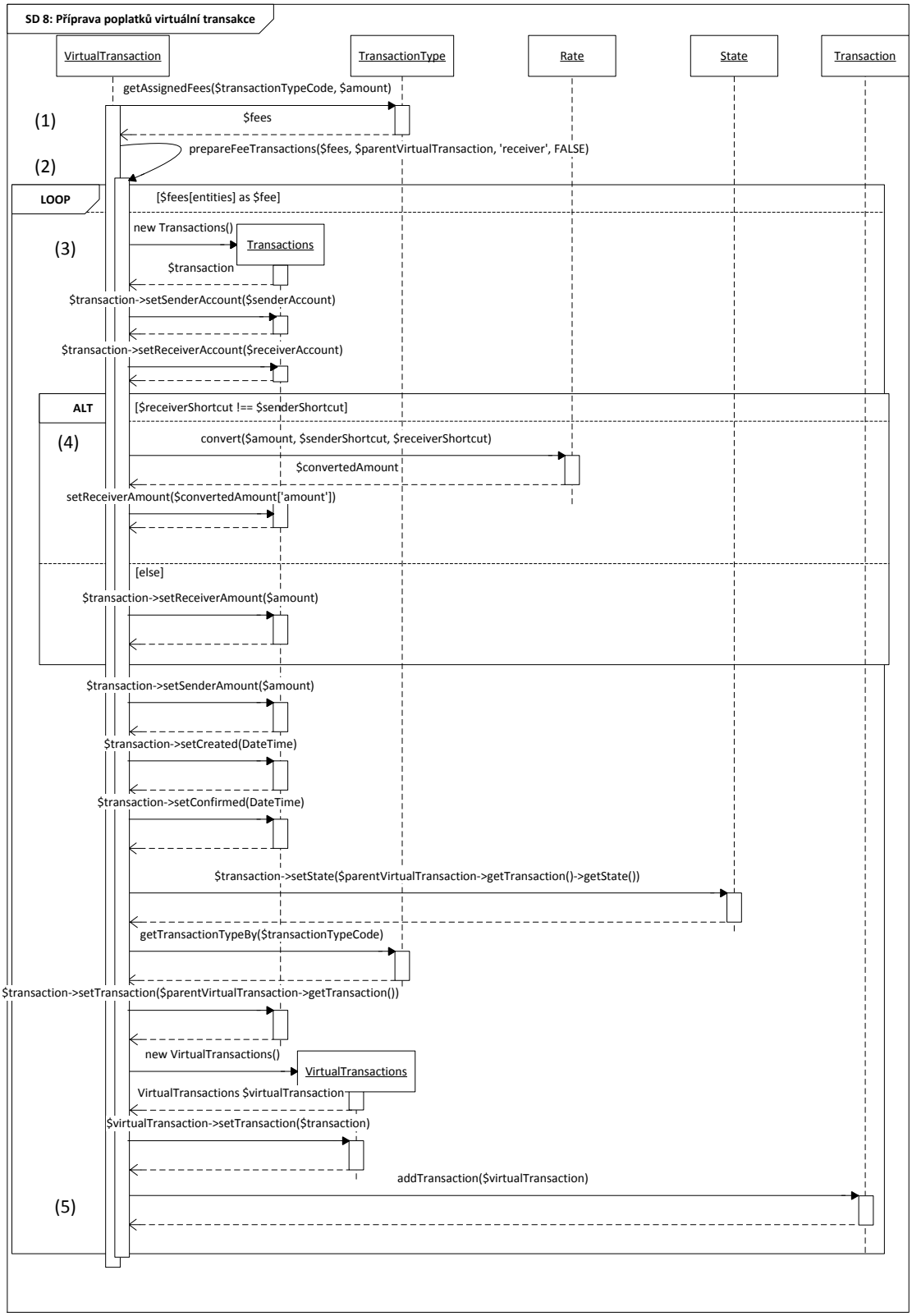
Obrázek 11: SD5: Příprava transakce Paypal poplatku



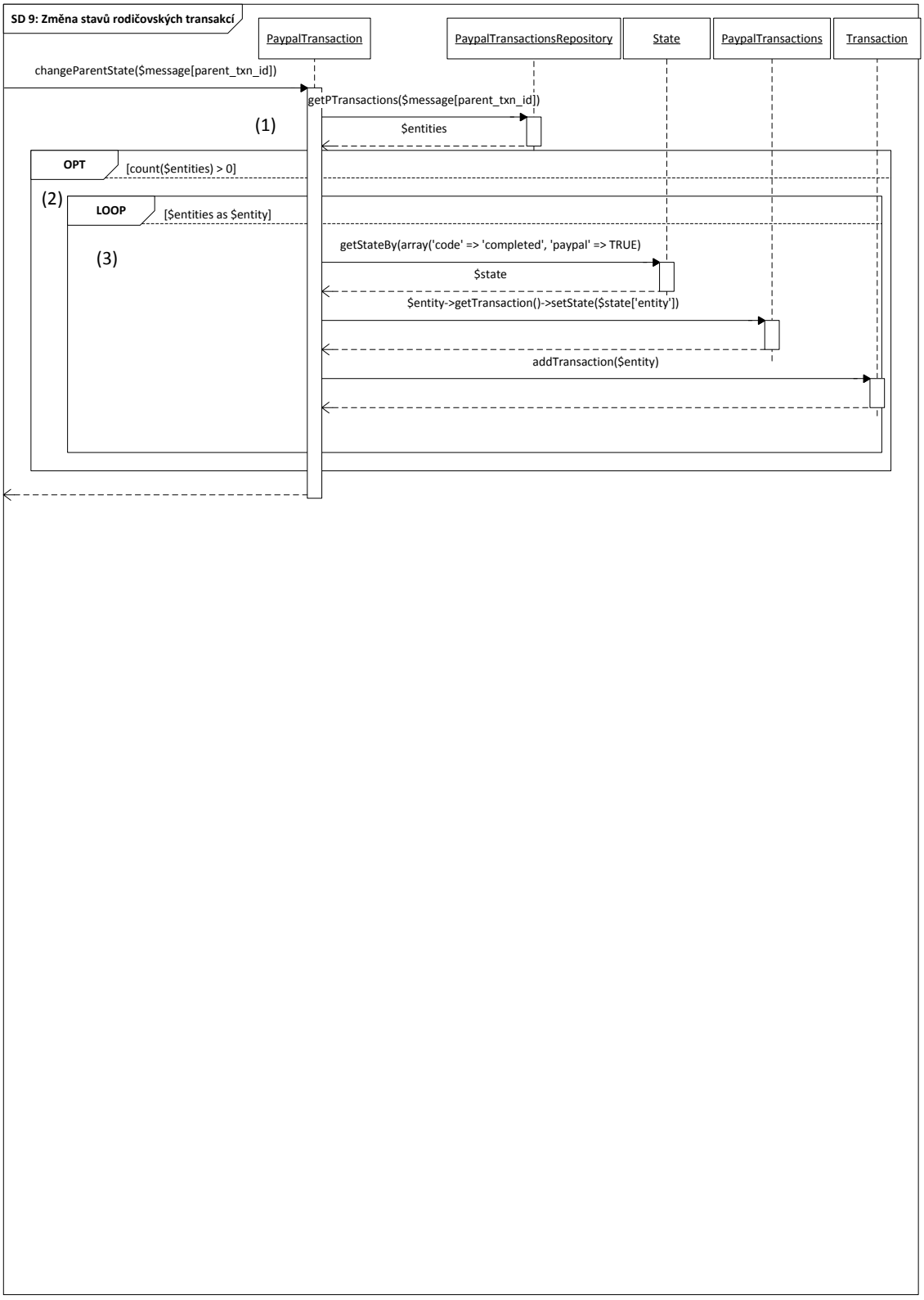
Obrázek 12: SD6: Provedení alternativního bloku



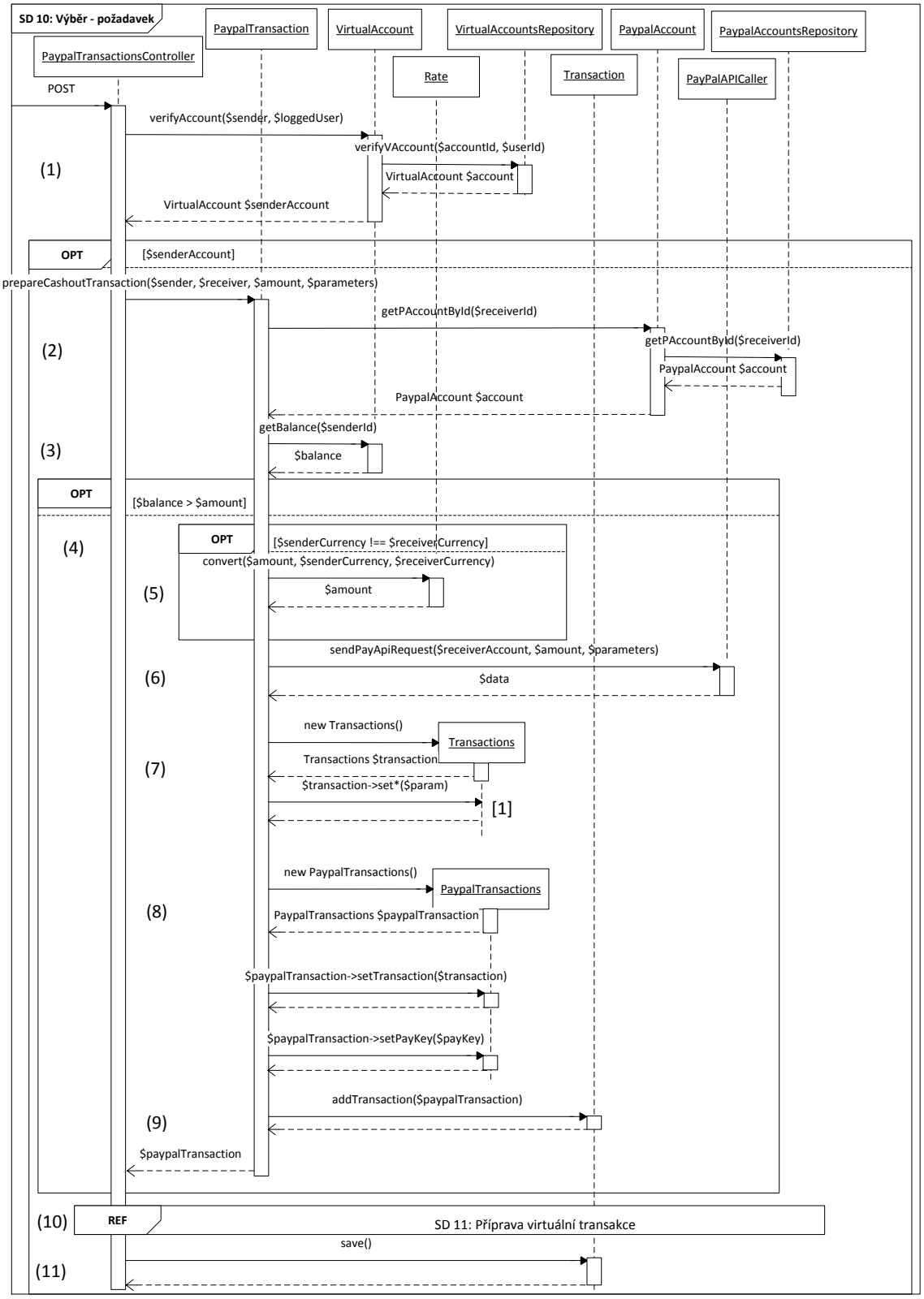
Obrázek 13: SD7: Příprava virtuální transakce



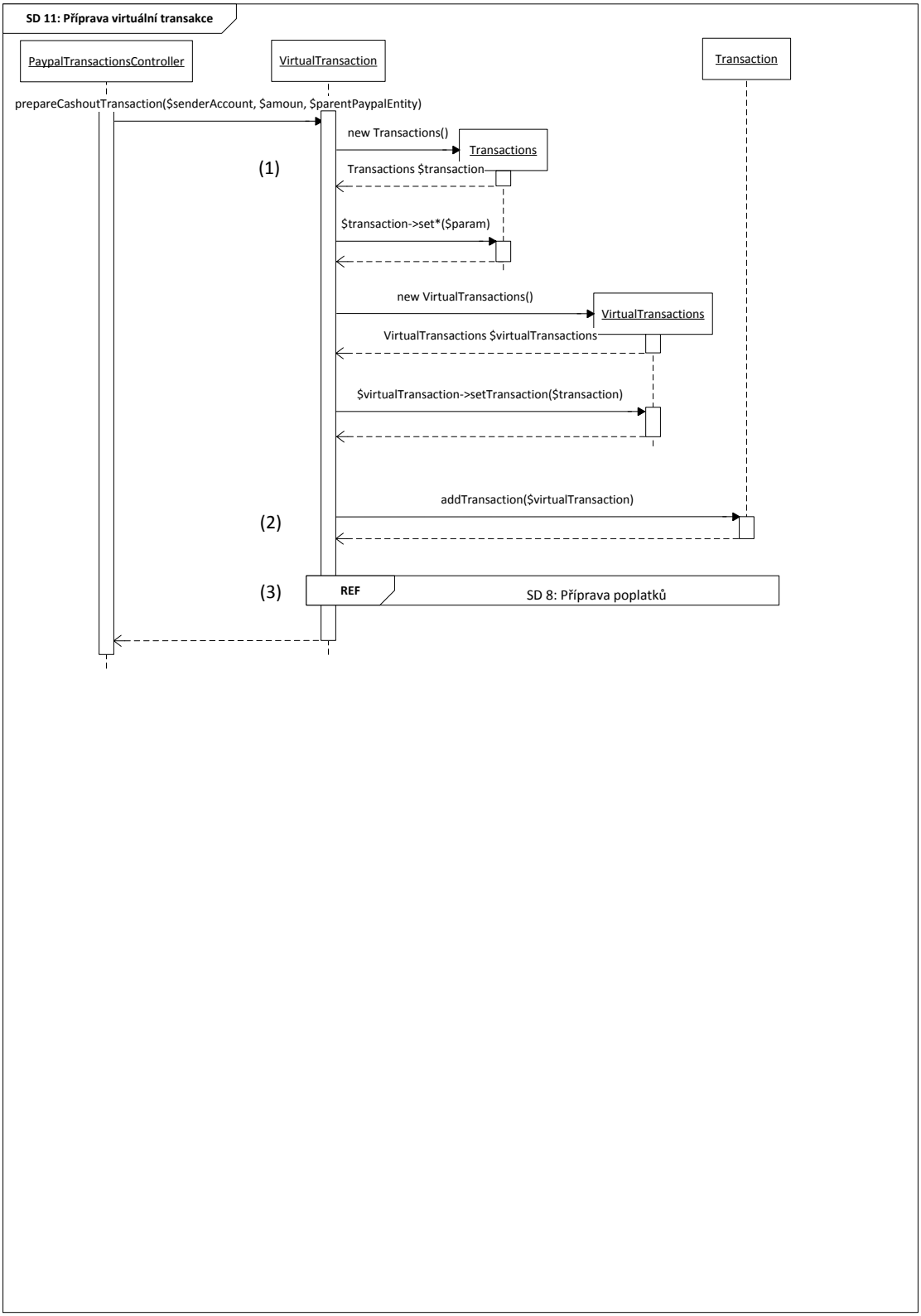
Obrázek 14: SD8: Příprava poplatků



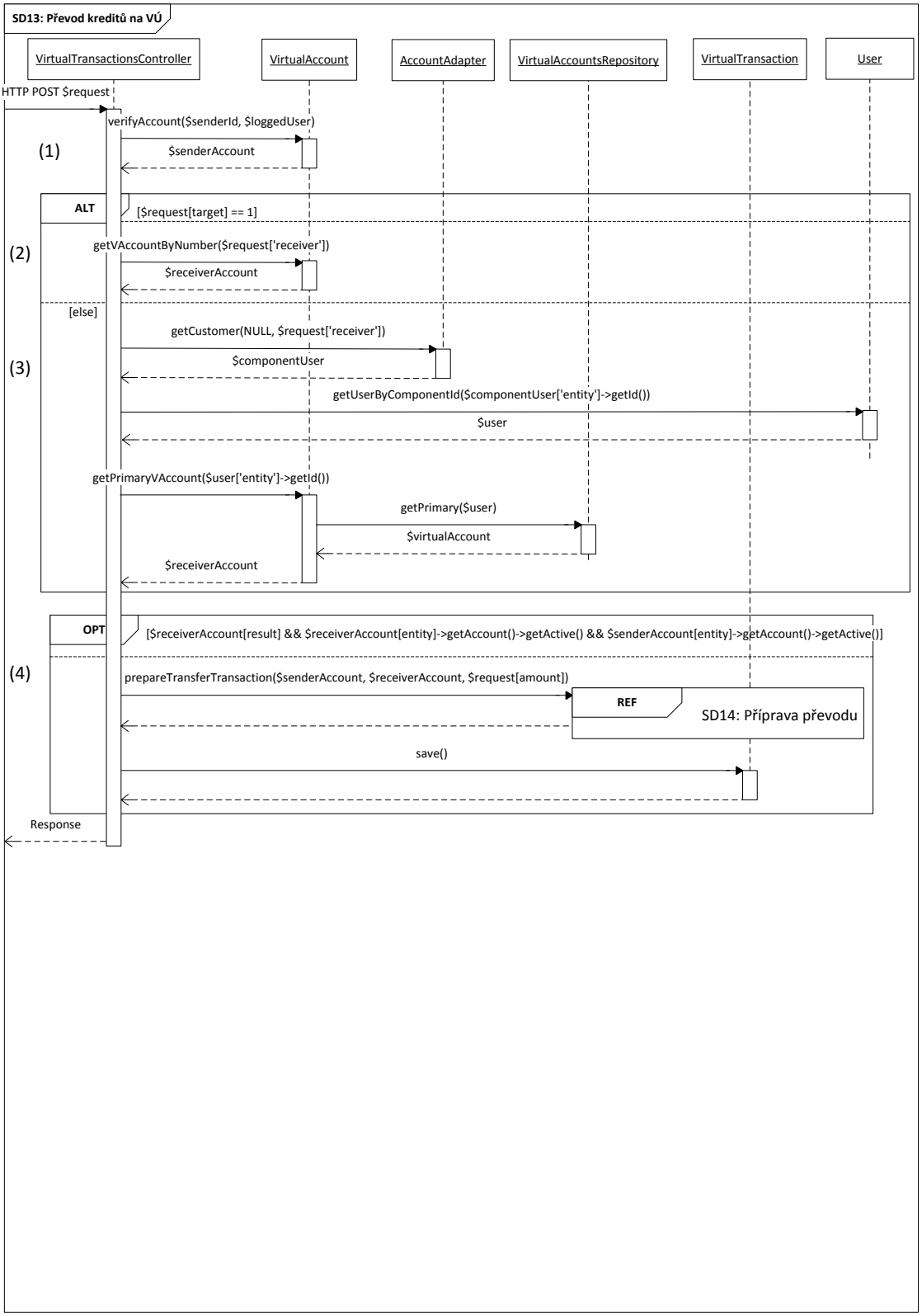
Obrázek 15: SD9: Změna stavů rodičovských transakcí



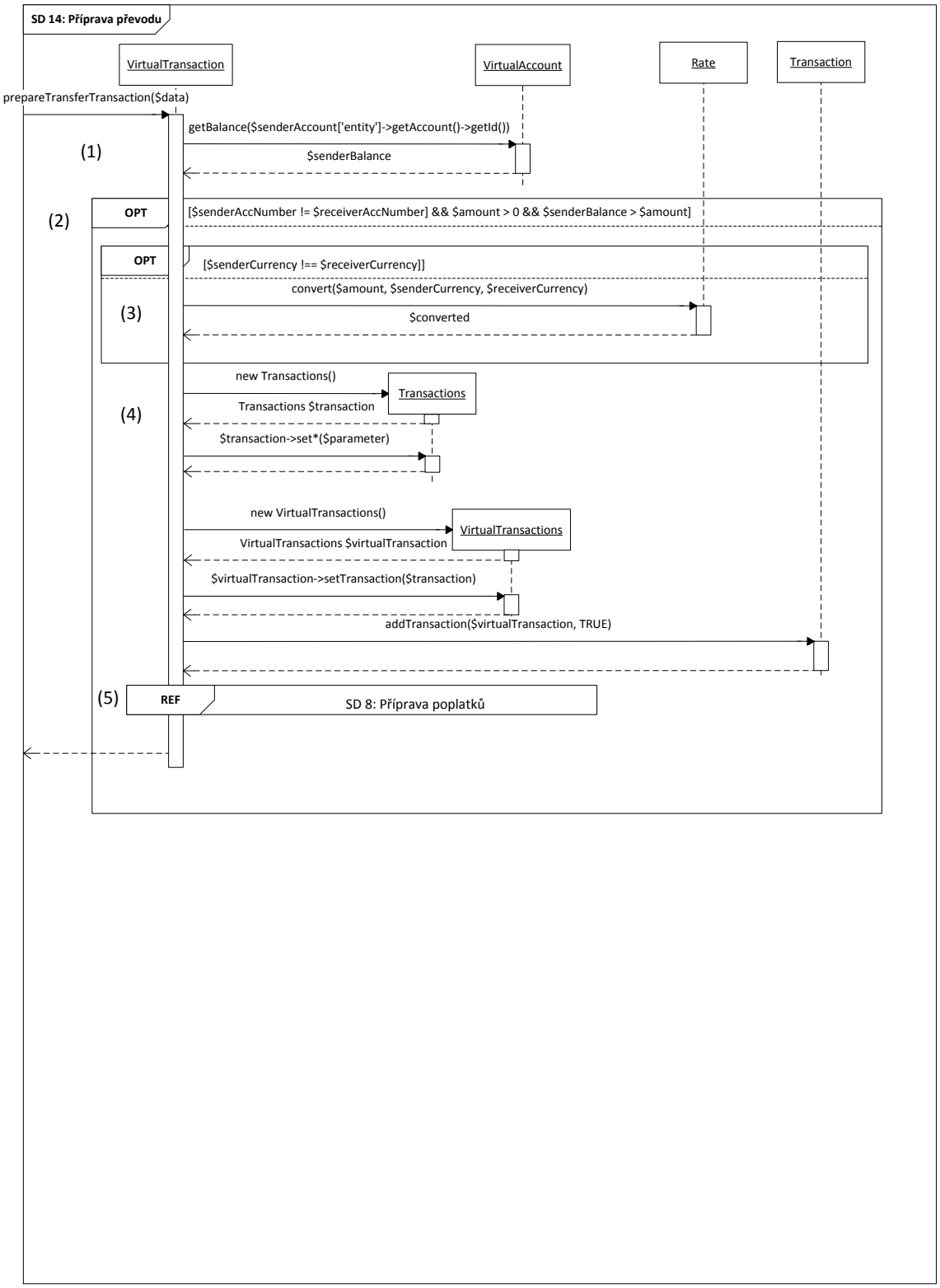
Obrázek 16: SD10: Výběr - požadavek



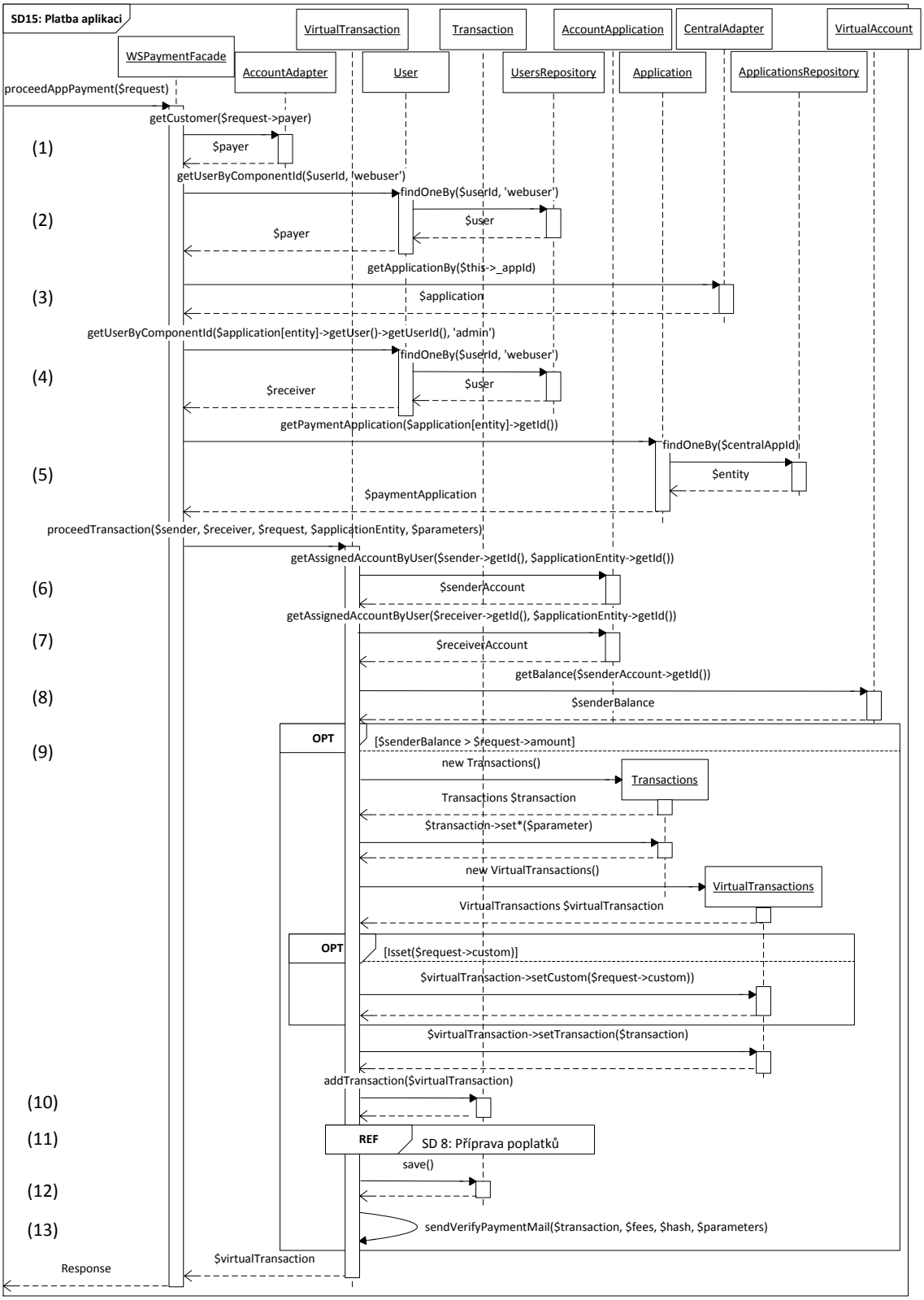
Obrázek 17: SD11: Příprava virtuální transakce



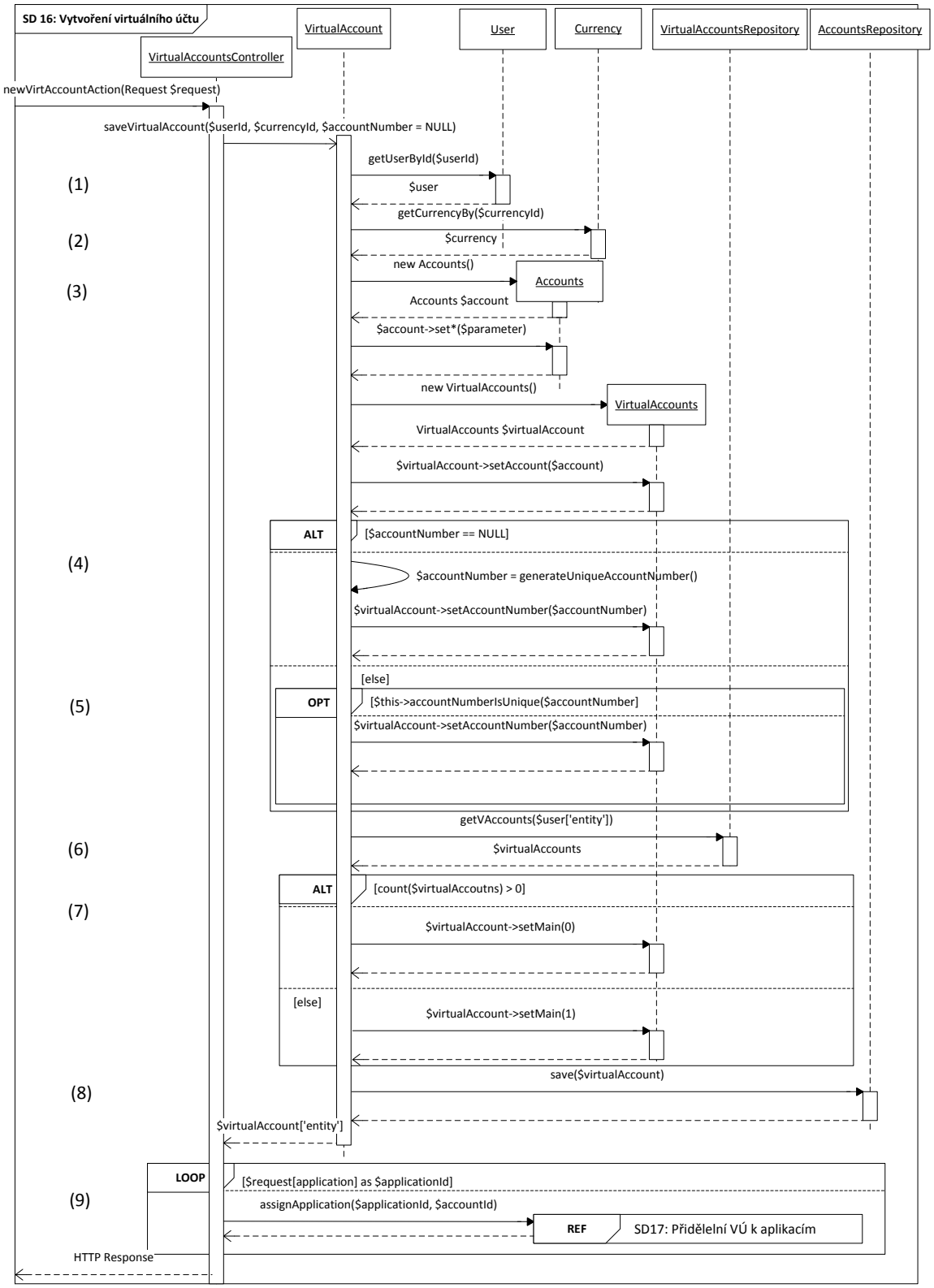
Obrázek 18: SD13: Převod kreditů na VÚ



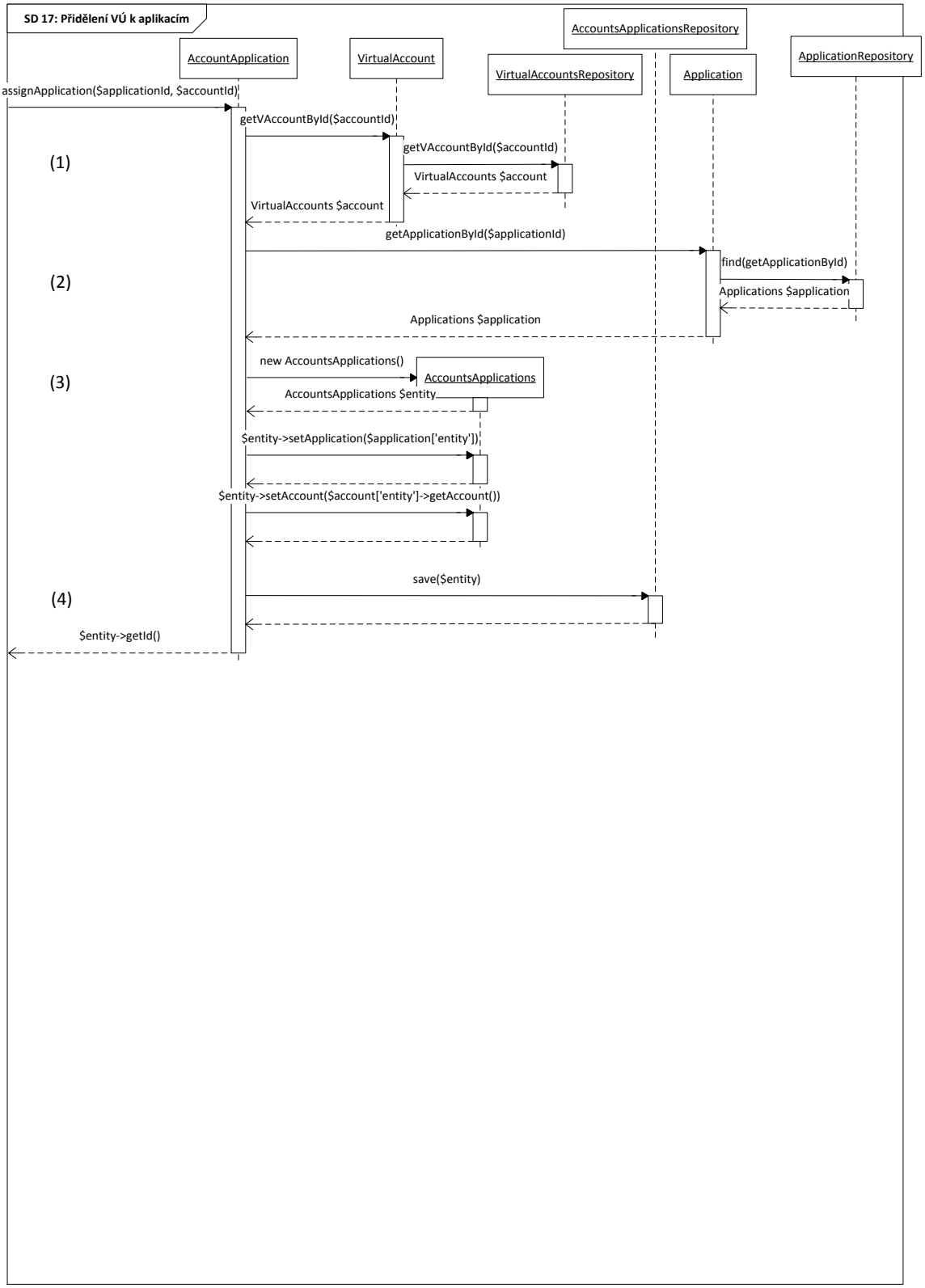
Obrázek 19: SD14: Příprava převodu



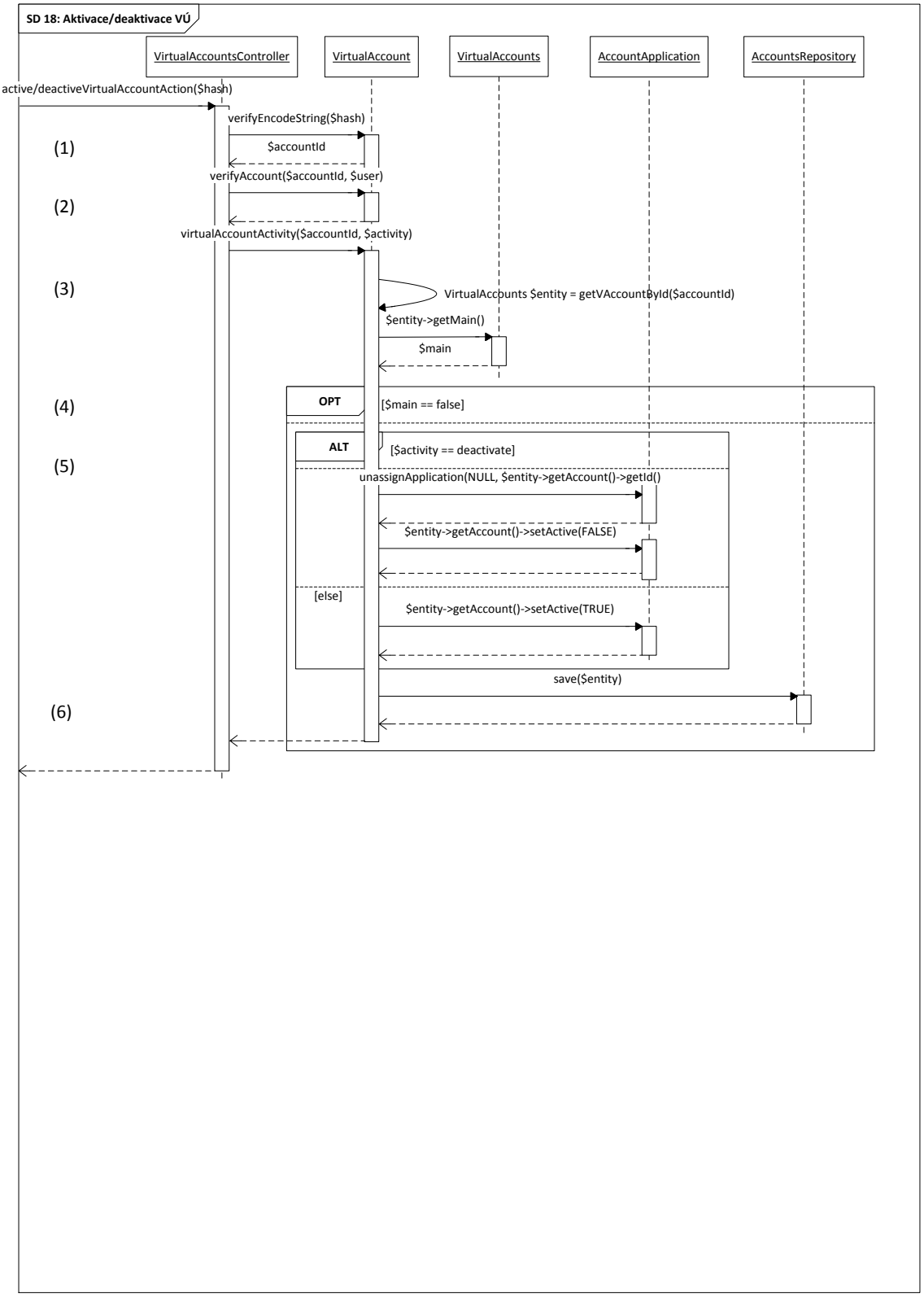
Obrázek 20: SD15: Platba aplikaci



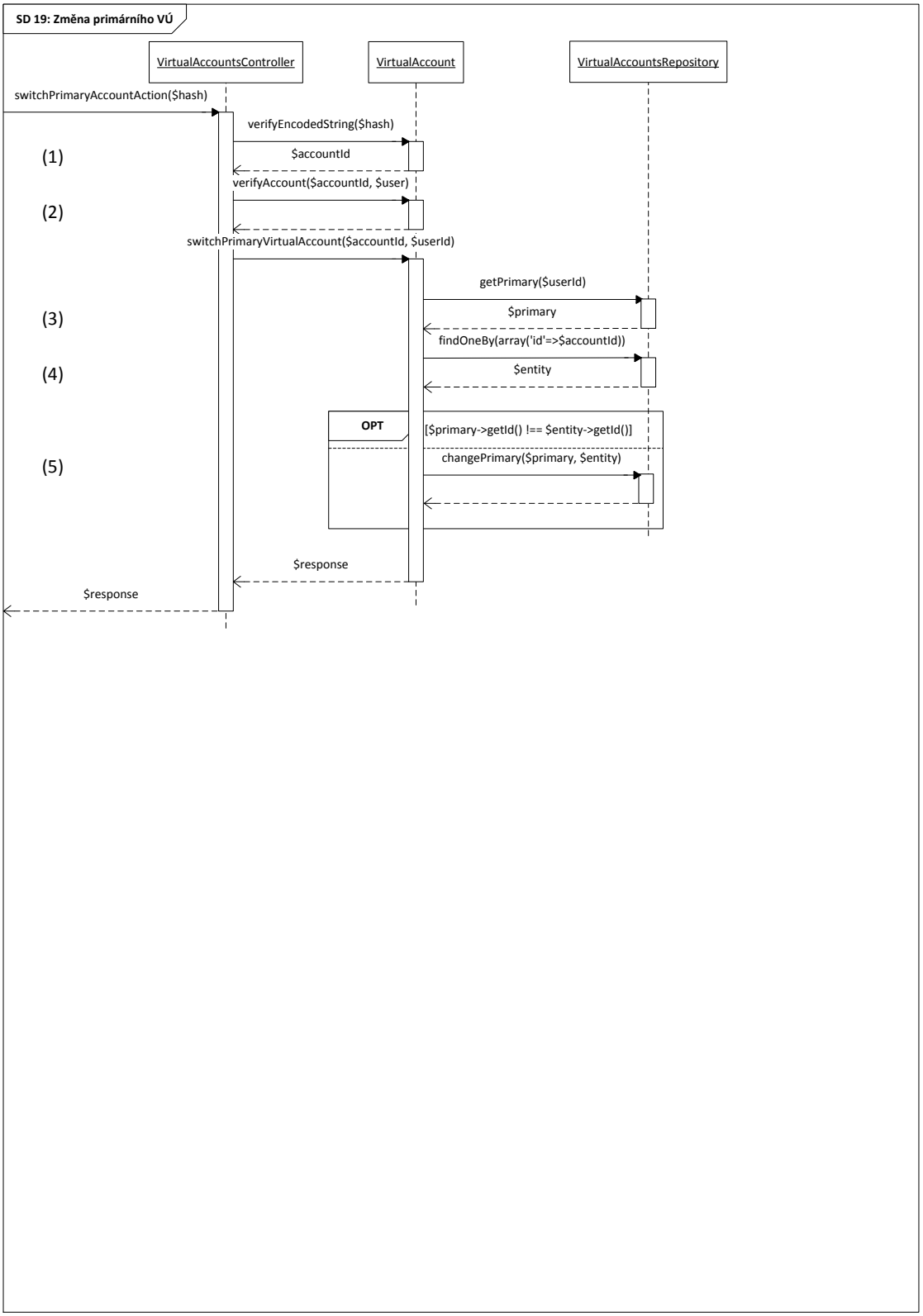
Obrázek 21: SD16: Vytvoření virtuálního účtu



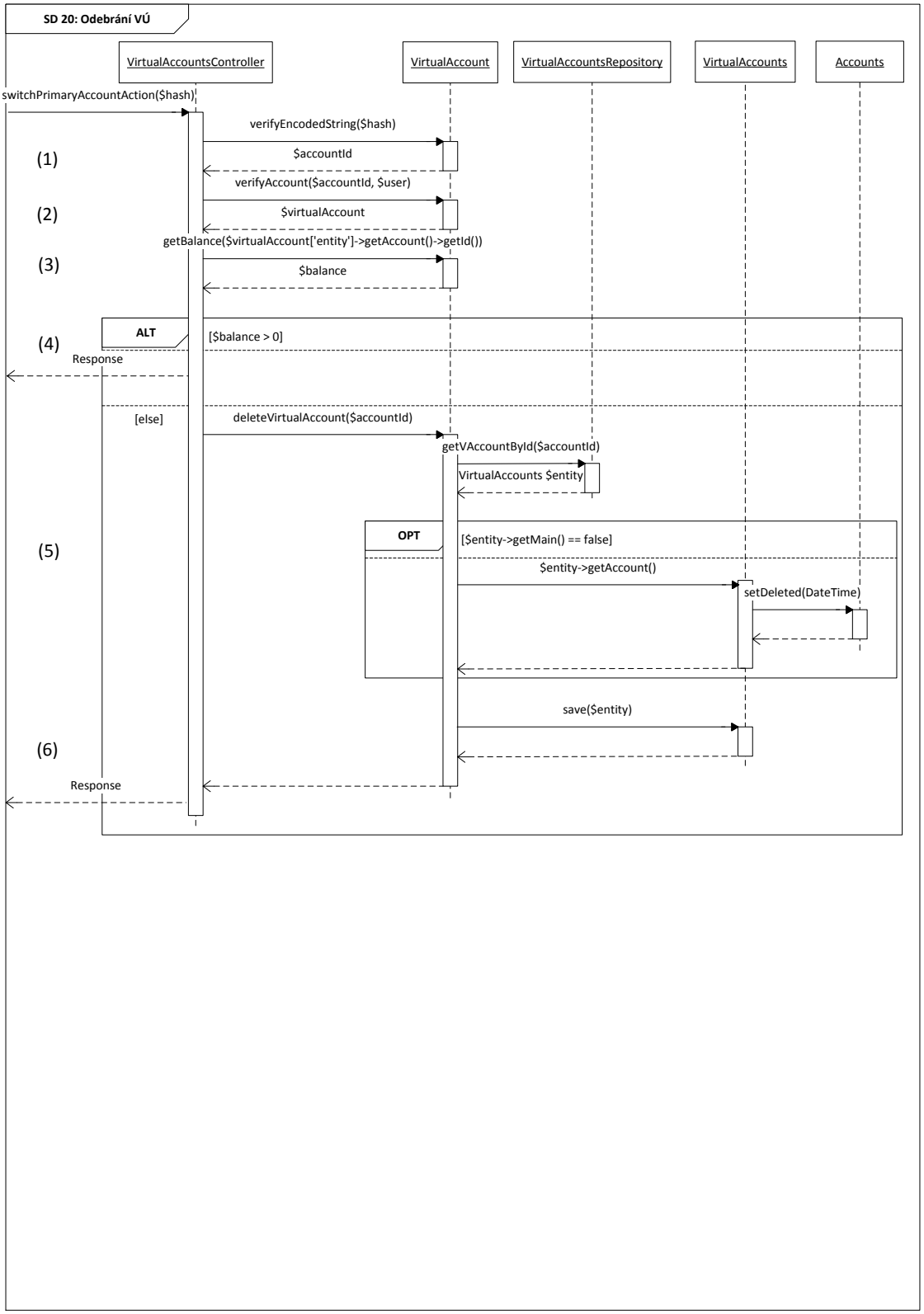
Obrázek 22: SD17: Přidělení VÚ k aplikacím



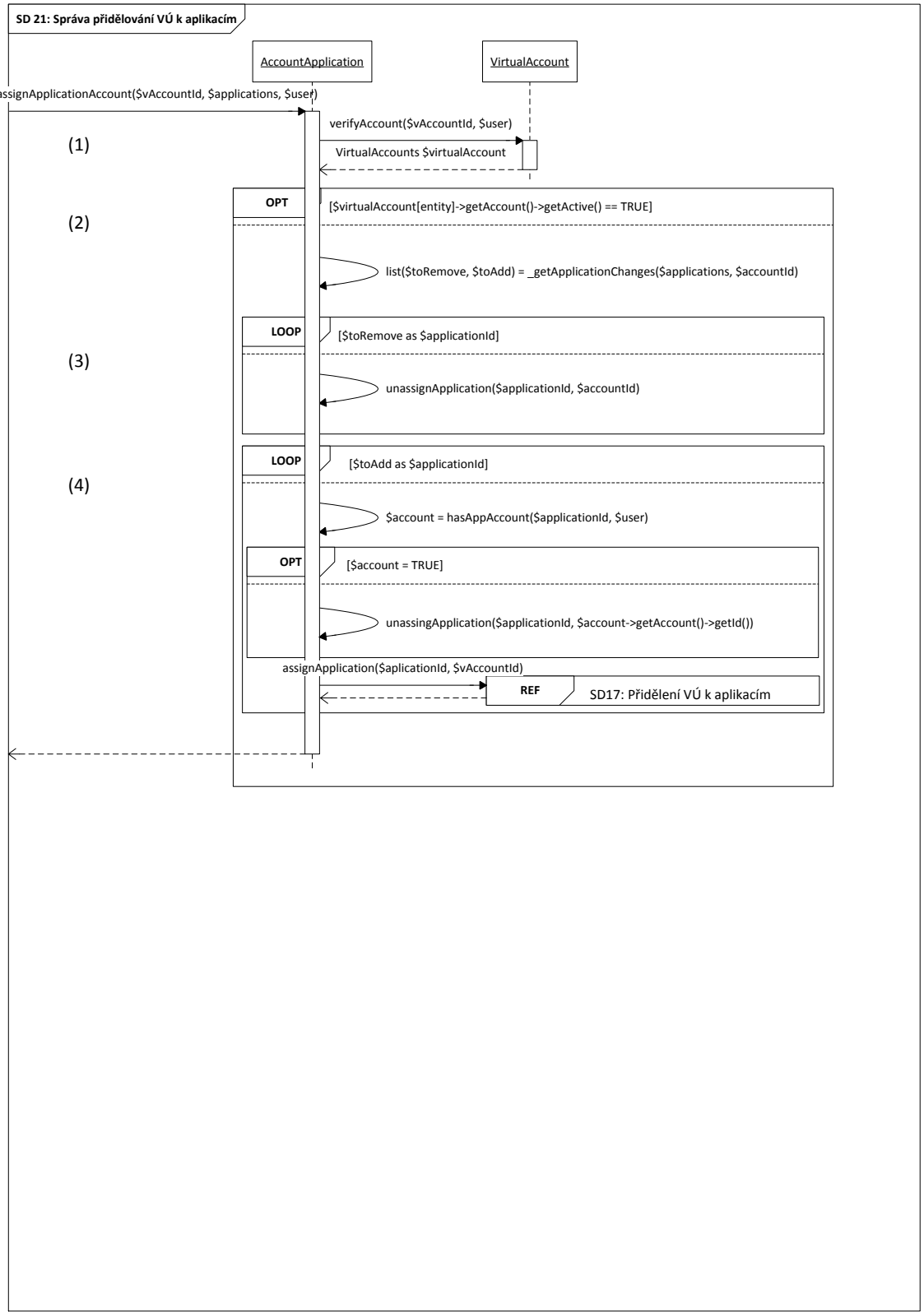
Obrázek 23: SD18: Aktivace/deaktivace VÚ



Obrázek 24: SD19: Změna primárního VÚ



Obrázek 25: SD20: Odebrání VÚ



Obrázek 26: SD21: Správa přidělování VÚ k aplikacím

D Scénáře případů užití

- **Případ užití:** PayPal výběr

- **Předpoklady:**
  1. Aktér je přihlášen do kreditního systému.
  2. Superadmin má evidován alespoň jeden PayPal účet.

- **Hlavní scénář:**
  1. Aktér klikne na odkaz "Výběr".
  2. Kreditní systém zobrazí formulář pro provedení výběru.
  3. Aktér zvolí výši výběru.
  4. Aktér zvolí virtuální účet, ze kterého budou kredity vybrány.
  5. Aktér zvolí paypal účet, určený k příjmu prostředků.
  6. Aktér potvrdí formulář.
  7. Kreditní systém ověří existenci zvoleného virtuálního účtu.
  8. Kreditní systém ověří, zda-li s účtem manipuluje jeho vlastník.
  9. Kreditní systém ověří existenci Paypal účtu.
  10. Kreditní systém ověří dostatečný zůstatek na virtuálním účtu.
  11. Pokud je měna virtuálního účtu odlišná od měny Paypal účtu, kredity se převedou na měnu Paypal účtu podle aktuálního kurzu.
  12. Kreditní systém odešle požadavek na systém Paypal, který bezproztředně vrátí klíč určený k potvrzení platby.
  13. Kreditní systém zaeviduje transakci včetně získaného klíče.
  14. Superadmin v centrálním systému klikne na tlačítko potvrdit platbu nesoucí klíč požadované transakce.
  15. Superadmin je přesměrován do rozhraní Paypal.
  16. Superadmin provede přihlášení a platbu potvrdí.
  17. PayPal o této skutečnosti informuje kreditní systém formou IPN zprávy.
  18. Kreditní systém edituje transakci a označí ji za dokončenou.

- **Případ užití:** Převody mezi účty (uživateli)

- **Předpoklady:**
  1. Aktér je přihlášen do kreditního systému.
  2. Aktér má vytvořen alespoň jeden virtuální účet.

- **Hlavní scénář:**
  1. Aktér vyplní všechna povinná data (vlastní virtuální účet, ze kterého budou kredity odeslány, cílový účet nebo e-mail příjemce a částku) a potvrdí formulář tlačítkem "Odeslat".
  2. Kreditní systém ověří, zda-li s účtem manipuluje jeho vlastník.
  3. POKUD s účtem nemanipuluje vlastník:
     (a) Kreditní systém o tomto informuje aktéra.
     (b) Proces převodu kreditů je neuspěšně ukončen.
  4. POKUD aktér zvolil možnost převodu na jiného uživatele, pokračuje se alternativním scénářem.
  5. Kreditní systém ověří, zda-li virtuální účet příjemce existuje.
  6. POKUD virtuální účet neexistuje:
     (a) Kreditní systém o tomto informuje aktéra.
     (b) Proces převodu kreditů je neuspěšně ukončen.
  7. Kreditní systém ověří, zda-li existuje virtuální účet plátce.
  8. POKUD virtuální účet neexistuje:
     (a) Kreditní systém o tomto informuje aktéra.
     (b) Proces převodu kreditů je neuspěšně ukončen.
  9. Kreditní systém ověří, zda-li účet příjemce není stejný s účte plátce.
  10. POKUD jsou účty stejné:
      (a) Kreditní systém o tomto informuje aktéra.
      (b) Proces převodu kreditů je neuspěšně ukončen.
  11. Kreditní systém zkontroluje, zda-li se plátce nesnaží převést zápornou nebo nulovou částku.
  12. POKUD ano:
      (a) Kreditní systém o tomto informuje aktéra.
      (b) Proces převodu kreditů je neuspěšně ukončen.
  13. Kreditní systém zda-li má plátce dostatečný zůstatek k provedení transakce.
  14. POKUD na účtu plátce není dostatečný zůstatek:
      (a) Kreditní systém o tomto informuje aktéra.
      (b) Proces převodu kreditů je neuspěšně ukončen.
  15. Kreditní systém dokončí proces převodu kreditů.
- **Alternativní scénář:**
  1. Podle zadaného e-mailu systém ověří, zda-li takový příjemce existuje.
  2. Pokud neexistuje:

   (a) Kreditní systém o tomto informuje aktéra.
   (b) Proces převodu kreditů je neuspěšně ukončen.
3. Kreditní systém zjistí číslo primárního účtu příjemce.
4. Scénář pokračuje krokem č.7

- **Případ užití:** Platba aplikaci
  Klient: klientská aplikace využívající kreditní systém
- **Předpoklady:**
  1. Aplikace má zaregistrován kreditní systém.
- **Hlavní scénář:**
  1. Klient zašle požadavek na kreditní systém obsahující povinné údaje, kterými jsou částka a token přihlášeného uživatele (plátce) a nepovinný údaj "custom".
  2. Kreditní systém ověří zda-li uvedený plátce skutečně existuje.
  3. POKUD takový uživatel neexistuje.
     (a) Kreditní systém vrátí klientovi hlášku o vzniklém problému.
  4. Kreditní systém ověří, zda-li má aplikace právo k provedení platby.
  5. POKUD takový uživatel neexistuje:
     (a) Kreditní systém vrátí klientovi hlášku o vzniklém problému.
  6. POKUD se měny plátce a aplikace liší, částka plátce je převedna pomocí aktuálního kurzu ČNB.
  7. Kreditní systém zaeviduje platební transakci.
  8. Kreditní systém zaeviduje poplatky k této transakci.

- **Případ užití:** Vytvoření nového virtuálního účtu
- **Předpoklady:**
  1. Aktér je přihlášen do kreditního systému.
- **Hlavní scénář:**
  1. Aktér klikne na odkaz „Vytvořit nový virtuální účet".
  2. Kreditní systém zobrazí formulář pro vytvoření nového virtuálního účtu.
  3. Aktér si zvolí, zda-li zadá vlastní číslo účtu nebo si nechá číslo vygenerovat.
  4. POKUD požaduje zadání vlastního čísla účtu:
     (a) Kreditní systém umožní aktérovi zadat vlastní číslo účtu.
     (b) Aktér zvolí vlastní číslo účtu (9-ti místné číslo).
  5. Aktér si zvolí měnu, ve které bude virtuální účet provozovat.

6. Aktér zvolí aplikace, ke kterým se má účet přiřadit (nepovinné).
7. Aktér potvrdí formulář.
8. Kreditní systém ověří zda-li jsou vstupní data validní.
9. POKUD vstupní data nejsou validní pokračuje alternativní scénář.
10. Kreditní systém ověří dostupnost jiných účtů aktéra.
11. POKUD aktér nemá další virtuální účty, systém určí nově vytvářený účet jako primární.
12. Kreditní systém dokončí proces vytvoření virtuálního účtu a informuje aktéra o úspěšném dokončení.

- **Alternativní scénář:**

1. Kreditní systém znovu zobrazí formulář spolu s hlášením o chybách, čímž informuje akterá, kde se stal problém.
2. POKUD aktér požaduje opravu vstupních dat, pokračuje se krokem 3 hlavního scénáře.

- **Případ užití:** Přiřazení/odebrání účtu aplikacím
- **Předpoklady:**

1. Aktér je přihlášen do kreditního systému.
2. Aktér má vytvořen alespoň jeden virtuální účet.

- **Hlavní scénář:**

1. Aktér si vybere účet, který si přeje editovat a klikne na ikonu s titulkem „Konfigurace účtu“ náležící danému účtu.
2. Kreditní systém ověří, zda-li s účtem manipuluje jeho vlastník.
3. POKUD s účtem nemanipuluje vlastník:
   (a) Kreditní systém o tomto informuje aktéra.
   (b) Proces přiřazení/odebrání účtu aplikacím je neuspěšně ukončen.
4. Kreditní systém přesměruje aktéra do rozhraní pro konfiguraci účtu.
5. Aktér provede požadované změny v přiřazení a potvrdí formulář.
6. Kreditní systém ověří, zda-li je přiřazovaný účet aktivní.
7. POKUD účet aktvní není:
   (a) Kreditní systém o tomto informuje aktéra.
   (b) Proces přiřazení/odebrání účtu aplikacím je neuspěšně ukončen.
8. POKUD došlo k přiřazení nových aplikací:
   (a) Kreditní systém ověří, zda-li nově přiřazené aplikace nemají již přiřazen jiný virtuální účet.

   (b) POKUD ano:
      i. Kreditní systém ověří, zda-li tyto účty nemají u dané aplikace blokované kredity.
      ii. POKUD alespoň jeden účet blokované kredity má, kreditní systém ukončí proces přiřazení účtu k aplikacím a informuje o tom aktéra.
      iii. V opačném případě kreditní systém původně přiřazené účty od aplikací odebere a přiřadí jim aktérem nově zvolený účet.
9. POKUD došlo k odebrání aplikací:
   (a) Kreditní systém ověří, zda-li tento účet má u dané aplikace blokované kredity.
   (b) POKUD ano, kreditní systém ukončí proces odebrání účtu k aplikacím a informuje o tom aktéra.
   (c) V opačném případě kreditní systém odebere aplikacím tento účet.
10. Proces přiřazení/odbrání účtu k aplikacím je úspěšně dokončen.

- **Případ užití:** Odebrání virtuálního účtu
- **Předpoklady:**
  1. Aktér je přihlášen do kreditního systému.
  2. Aktér má vytvořen alespoň jeden virtuální účet.
- **Hlavní scénář:**
  1. Aktér si vybere účet, který si přeje smazat a klikne na ikonu s titulkem „Smazat“ náležící danému účtu.
  2. Kreditní systém požádá o potvrzení odebrání virtuálního účtu.
  3. POKUD aktér tuto operaci zruší, akce je ukončena.
  4. Kreditní systém ověří, zda-li s účtem manipuluje jeho vlastník.
  5. POKUD se nejedná o vlastníka:
     (a) Kreditní systém upozorní aktéra o tom, že tento virtuální účet nemůže odebrat.
     (b) Operace odebrání účtu je neúspěšně ukončena.
  6. Kreditní systém ověří, zda-li se nejedná o primární účet.
  7. POKUD se jedná o primární účet:
     (a) Kreditní systém upozorní aktéra, že není možné odebrat primární účet.
     (b) Operace odebrání účtu je neúspěšně ukončena.
  8. Kreditní systém zkontroluje, jestli jsou na účtu uloženy kredity.
  9. POKUD na účtu kredity nejsou, pokračuje se alternativím scénářem.
  10. Kreditní systém upozorní aktéra a zároveň mu nabídne, zda-li nechce kredity převést na svůj primární účet a následně požadovaný účet odebrat.

11. POKUD aktér tento dotaz nepotvrdí:
    (a) Kreditní systém neprovede žádnou operaci a proces smazání virtuálního účtu tímto skončil.
12. Kreditní systém zkontroluje, zda-li na účtu nejsou blokované kredity.
13. POKUD ano:
    (a) Kreditní systém o této skutečnosti informuje aktéra.
    (b) Operace odebrání virtuálního účtu je neúspěšně ukončena.
14. Kreditní systém odebere účet všem přiřazeným aplikacím.
15. Kreditní systém převede kredity na primární účet a požadovaný účet smaže.
16. Operace odebrání virtuálního účtu je úspěšně dokončena.

- **Alternativní scénář:**

1. Kreditní systém odebere tento účet všem přiřazeným aplikacím.
2. Kreditní systém smaže požadovaný účet.
3. Operace odebrání virtuálního účtu je úspěšně dokončena.

- **Případ užití:** Deaktivace virtuálního účtu
- **Předpoklady:**

1. Aktér je přihlášen do kreditního systému.
2. Aktér má vytvořen alespoň jeden virtuální účet.

- **Hlavní scénář:**

1. Aktér si vybere účet, který chce deaktivovat a klikne na ikonu s titulkem „Deaktivovat účet“ náležící k danému účtu.
2. Kreditní systém požádá o potvrzení, zda-li chce aktér doopravdy účet deaktivovat.
3. POKUD aktér ověření potvrdí:
    (a) Kreditní systém ověří, zda-li s účtem manipuluje jeho vlastník.
    (b) POKUD se nejedná o vlastníka:
        i. Kreditní systém upozorní aktéra, o tom, že tento virtuální účet nemůže odebrat.
        ii. Operace odebrání účtu je neúspěšně ukončena.
    (c) Kreditní systém ověří, zda-li na účtu nejsou blokované kredity.
    (d) POKUD ano:
        i. Kreditní systém o této skutečnosti informuje aktéra.
        ii. Operace odebrání virtuálního účtu je neúspěšně ukončena.
    (e) Kreditní systém odebere účet všem přiřazeným aplikacím.

(f) Kreditní systém deaktivuje virtuální účet.

4. Operace deaktivace virtuálního účtu je úspěšně dokončena.

- **Případ užití:** Aktivace virtuálního účtu
- **Předpoklady:**
  1. Aktér je přihlášený do kreditního systému.
  2. Aktér má vytvořen alespoň jeden virtuální účet.
- **Hlavní scénář:**
  1. Aktér si vybere účet, který chce aktivovat a klikne na ikonu s titulkem „Aktivovat účet“ náležící k danému účtu.
  2. Kreditní systém požádá o potvrzení, zda-li chce aktér doopravdy účet aktivovat.
  3. POKUD aktér ověření potvrdí:
     (a) Kreditní systém ověří, zda-li s účtem manipuluje jeho vlastník.
     (b) POKUD se nejedná o vlastníka.
        i. Kreditní systém upozorní aktéra, o tom, že tento virtuální účet nemůže odebrat.
        ii. Operace odebrání účtu je neůspěšně ukončena.
     (c) Kreditní systém aktivuje účet.
  4. Operace aktivace virtuálního účtu je úspěšně dokončena.
- **Případ užití:** Změna primárního účtu
- **Předpoklady:**
  1. Aktér je přihlášen do kreditního systému.
  2. Aktér má vytvořen alespoň jeden virtuální účet.
- **Hlavní scénář:**
  1. Aktér si vybere virtuální účet a klikne na ikonu s titulkem „Nastavit jako primární účet“ náležící danému účtu.
  2. Kreditní systém požádá o potvrzení, zda-li aktér skutečně požaduje přepnout primární účet.
  3. POKUD aktér ověření potvrdí:
     (a) Kreditní systém ověří, zda-li s účtem manipuluje jeho vlastník.
     (b) POKUD se nejedná o vlastníka.
        i. Systém upozorní aktéra, o tom, že tento virtuální účet nemůže odebrat.
        ii. Operace odebrání účtu je neůspěšně ukončena.

(c) Kreditní systém ověří, zda-li je daný účet aktivní.
(d) POKUD aktivní není:
    i. Kreditní systém daný účet aktivuje.
(e) Kreditní systém přepne primární účet.

4. Operace změny primárního virtuálního účtu je úspěšně dokončena.